Disseny i desenvolupament d'un videojoc

Genís Bayona i Jaume

Treball de final de grau
del grau en enginyeria informàtica

Facultat Informàtica de Barcelona
Universitat Politècnica de Catalunya

Tutor: Guillem Godoy Balil

Octubre 2016

Resum

Aquest projecte és la culminació del meu pas pel grau en enginyeria informàtica de la FIB. Per tant, és un projecte de gran importància tant a nivell personal com professional, i en el que he de reflectir el meu aprenentatge d'aquests darrers anys. La meva decisió per a tal fí ha estat dissenyar i programar un videojoc.

El disseny del projecte, ha vist la llum després d'un intens treball de creativitat i anàlisi d'alternatives per les mecàniques, estil artístic i experiència de joc. El disseny final s'ha consolidat després de fer un estudi del mercat actual i històric dels videojocs.

El codi del projecte ha estat desenvolupat sobre les bases d'Escalabilitat, Usabilitat, Reusabilitat i Eficiència. Les parts més concretes del joc s'han treballat de forma específica per solucionar les necessitats puntuals. Pels sistemes més generals s'ha treballat de forma més abstracta i genèrica per tal de donar la possibilitat de ser reaprofitats en futurs projectes.

Per la programació he utilitzat el llengüatge de programació C++, i les llibreries multimèdia de SFML (Simple and Fast Multimedia Library) que m'ha permès programar el projecte a bastant baix nivell.

El joc tracta d'un personatge al qual una misteriosa ma robòtica ha robat el barret. Per tal de recuperar-lo, s'endinsarà en una cova amb estranys escrits a les parets en la qual anirà trobant barrets i lluitant contra els enemics disparant-los i utilitzant un ganxo extensor per moure's pel nivell. Si arriba al final, potser recuperarà allò que hi ha anat a buscar, però el que és segur, és que la cova no li posarà fàcil.

Abstract

This is the final project of my computer science degree, which I am completing at the FIB. For this reason, this project has a huge importance both personally and professionally, and is designed to reflect what I have learned during my studies. To do so, I have decided to design and develop a video game.

The design of the project was first presented after much work was put into the creativity and analysis of alternatives for the mechanics, graphical style and gameplay experience. The final design was a result of a study about the current and past market of video games.

The code of the project was developed using ideas of Scalability, Usability, Reusability and Efficiency. The most concrete parts have been developed in such a way that they solve the specific needs based purely on this game. For the more general systems, an abstract and generic approach has been taken instead. This way, the code for general problems can be reused in future projects.

For developing the game I used C++ as the programming language and SFML (Simple and Fast Multimedia Library), which allowed me to use low level systems.

The game is about a character whose hat was stolen by a mysterious robotic hand. To recover it, the character will enter an unknown cave with strange paintings on the walls where they will find hats, fight the enemies shooting at them and use a grappling hook to move around the room. If the last room of the cave is reached the stolen hat may be recovered, but the cave may not make it easy to get it back.

Resumen

Este proyecto es la culminación de mi paso por el grado en ingeniería informática en la FIB. Por eso, es un proyecto de gran importancia tanto a nivel personal como profesional, y en el que se tiene que ver reflejado mi aprendizaje a lo largo de estos últimos años. La decisión para tal fin, ha sido diseñar y programar un videojuego.

El diseño del proyecto, vió la luz después de un intenso trabajo de creatividad y análisis de alternativas para las mecánicas, estilo artístico y experiencia de juego. El diseño final, se consolidó después de un estudio del mercado actual e histórico de los videojuegos.

El código del proyecto ha sido desarrollado sobre las bases de Escalabilidad, Usabilidad, Reusabilidad y Eficiencia. Las partes mas concretas del juego se han trabajado de forma específica par solucionar las necesidades puntuales. Para los sistemas mas generales, se ha trabajado de forma mas abstracta y genérica para dar la posibilidad de ser reutilizadas en futuros proyectos.

Para la programación he utilizado el lenguaje de programación C++, y las librerías multimedia de SFML (Simple and Fast Multimedia Library) que me ha permitido programar el proyecto a bastante bajo nivel.

El juego trata de un personaje al cual una misteriosa mano robótica ha robado el sombrero. Para recuperarlo, se adentrará en una cueva con extraños escritos en las paredes de la cual irà encontrando sombreros y luchando contra los enemigos disparando y usando su gancho extensor para moverse por el nivel. Si llega al final, quizás pueda recuperar lo que anda buscando, pero lo que si es seguro es que la cueva no se lo va a poner fácil.

Al llarg del desenvolupament del projecte, s'ha anat actualitzant una pàgina web amb les novetats. D'aquesta manera tothom que hi estigués interessat podia veure el progrés i tot allò que s'anava afegint. A més a més, també serveix com a presentació del joc, ja que les imatges que s'hi mostren ensenyen escenes reals, i permeten que el jugador es faci una idea de com serà. Per accedir a la pàgina es pot utilitzar el següent enllaç **www.whotookmyhat.webs.com**. A l'apartat de descàrregues es pot trobar un executable per windows, i l'enllaç al github on es pot trobar el codi font.

Índex

Capítol 1

Introducció

1.1 Context

L'indústria dels videojocs ha tingut un comportament de creixement continuat des dels seus inicis. Es tracta d'un món viu, molt actiu, amb grans canvis i molts actors. Però no només mou grans quantitats de diners. A través de l'oferta d'experiències, també s'aconsegueix moure quelcom que és vital, que és la vida dels seus jugadors.

Els primers videojocs, o prototips de videojocs, construïts sobre oscil·loscopis daten d'entre 1952 (TIC TAC TOE de A.S. Douglas) i 1958 (Tennis for Two de William Higinbotham), dues dates sempre en discussió a l'historia dels videojocs. Però això no va ser més que el tret de sortida. A partir d'aquells primers conceptes de joc, s"han vist videojocs de tots els tipus i en tots els formats. Des dels Arcades i primeres consoles dels anys 70, als primers ordinadors personals de la dècada dels 80, les consoles portables que van entrar en joc al voltant dels 90, el gran canvi social i tecnològic de les plataformes de mòbil i el creixement del web i l'entreteniment online del 2000, i les perspectives dels sistemes de Realitat augmentada i Realitat Virtual que ja han posat mig peu al mercat. Amb l'avenç de la tecnologia també ha tingut lloc el creixement i la consolidació del concepte indie: a partir d'aquest moment ja no només les empreses amb gran capital poden fer jocs, doncs les consoles o sistemes operatius comencen a oferir noves formes de desenvolupar per a la seva plataforma, i la tecnologia cada vegada més accessible permet que qualsevol pugui entrar en aquest món, i que el mercat creixi i s'expandeixi. Aquest mercat també es diversifica, doncs es comencen a fer jocs per a sectors que no havien estat mai el públic objectiu dels videojocs, amb un gran èxit.

En aquest mapa de gran activitat trobem també gran competitivitat. És cert que és més fàcil entrar en el sector, però també més difícil destacar-hi. Ha crescut la demanda, i per bé que les facilitats per desenvolupar han fet fàcil l'entrada, també ha donat lloc a una gran quantitat de productes al mercat, sovint de mala qualitat, i amb conceptes repetits o directament copiats d'altres jocs. En aquest context apareix la necessitat de crear jocs originals, que el públic pugui apreciar per la seva autenticitat i singularitat, o perquè hi ha trobat aquella experiència que busca i que no li han ofert els altres milers de jocs del mercat.

Per tal d'acabar de contextualitzar el projecte, més endavant veurem algunes classificacions dins de les que es podria incloure el joc a desenvolupar, i un anàlisi del mercat sobre jocs similars.

1.2 Formulació del problema

Aquest projecte és la culminació del meu pas pel grau en enginyeria informàtica de la FIB. Per tant, és un projecte de gran importància tant a nivell personal com professional, i en el que he de reflectir el meu aprenentatge d'aquests darrers anys. La meva decisió per a tal fí ha estat que el projecte consisteixi en el disseny i la programació d'un videojoc. El desenvolupament de videojocs requereix de bones habilitats de programació, utilització d'estructures de dades i algorismes eficients, i coneixements en gràfics i tecnologies diverses. A més a més, és un àmbit en el que estic especialment motivat, i de fet he creat alguns videojocs durant els meus estudis aprofitant el meu temps lliure, i també com a part del treball realitzat a les assignatures de la FIB d'Interacció i Disseny d'Interfícies, i de Videojocs. Com a incentiu afegit, val a dir que els videojocs són un camp d'actualitat en gran expansió, i que a més a més demana creativitat, atès que cada joc és un problema per si mateix que sovint dona lloc a solucions úniques i especials.

L'elecció del videojoc a desenvolupar ha tingut com a objectius principals, per una banda, poder posar a prova i reflectir els meus coneixements, i per altra banda crear un producte de qualitat per a l'usuari, que li permeti viure una experiència emocionant i interessant. El joc es pot emmarcar en diverses categories. Està dissenyat per a ordinadors personals, és local i mono-jugador. Des del punt de vista de la mecànica, el podem definir com un joc de plataformes 2D d'acció i tir. Gràficament combina diversos estils, de manera que també es pot classificar com un joc experimental. S'ha desenvolupat amb baix pressupost i està orientat a jugadors experimentats. En les següents seccions detallem tots aquests aspectes.

1.3 Descripció del joc per Classificacions

Classificar els videojocs ha estat sempre una tasca complicada degut a la seva varietat de mecàniques, temàtica, gràfics o estils, que sovint es combinen o tenen diferents nivells de detall. La millor forma d'entendre un joc és jugar-lo, perquè això ens dona informació rellevant difícil de categoritzar, i el nivell de detall descriptiu que només l'experiència pot oferir. Ens referim a aspectes com el nivell de dificultat, la jugabilitat, l'addicció i el nivell de satisfacció que provoca, i l'experiència de joc en general. Descriure de forma mínimament rigorosa els diferents tipus de videojoc que existeixen seria una tasca ingent i poc útil en el context d'aquest projecte. Per aquest motiu explicaré les categories en les que es troba el joc dins de les classificacions més comunes.

Ordinador personal:

El meu joc està pensat per executar-se i jugar-se des d'un Ordinador, tant sobretaula com portàtil. Per tant s'assumeix que s'utilitzarà amb una pantalla gran, i amb un control basat en teclat i ratolí, o amb comandament. Té pocs requeriments de hardware, però fa un ús considerable de memòria i capacitat de procés.

Joc 2D:

El joc funciona en un sistema de físiques de dues dimensions. Al seu torn, tots els elements gràfics es representen amb imatges també bidimensionals.

Plataformes:

Es pot considerar un joc de plataformes 2D, doncs essencialment movem un personatge sotmès a gravetat i amb la capacitat de saltar en un món en dues dimensions a on podem trobar plataformes suspeses a l'aire. Un exemple conegut d'aquest estil de joc és el Super Mario Bross.

Acció i Tir(Shooter):

La mecànica del joc emfatitza els reptes físics: posa a prova la capacitat d'anàlisi i reacció del jugador als esdeveniments del joc en el moment adequat i de la forma precisa. D'entre els videojocs d'acció, entraria dins la categoria dels jocs de tir (shooters), ja que el personatge pot i haurà de disparar bales per tal d'eliminar als enemics i superar els nivells.

Nivell de dificultat mig/alt:

Es tracta d'un joc amb unes mecàniques complicades, que requereixen de l'habilitat i l'atenció constant del jugador. La dificultat augmenta a mesura que es van superant nivells, requerint així que el jugador no perdi en cap moment l'atenció, i forçant-lo a millorar per tal de poder arribar al final. Val a dir que el jugador no haurà de completar novament aquells nivells que hagi superat en partides anteriors, doncs el propi sistema guarda memòria del progrés assolit, fent així que la dificultat sigui menor. Per la naturalesa del joc, els meus principals usuaris objectiu seran persones d'entre 12 i 35 anys que siguin jugadors de videojocs habituals, ja que són les generacions que han

crescut dins el món dels videojocs, i que ja han assolit prou coordinació i capacitat de raonament i reacció per jugar a aquest tipus de jocs en experiències prèvies. Dit això, vull remarcar que el meu objectiu no és excloure ningú de l'experiència i que el rang d'edat és intuïtiu i aproximat.

Joc Indie:

El desenvolupament ha comptat amb poc suport econòmic i un equip reduït. Això ha comportat una experiència de joc és senzilla i un contingut limitat, per bé que podem observar factors experimentals al llarg del joc, destaca de forma especial l'apartat artístic. Lluny de ser un producte empresarial de gran abast pot ser considerat una producció Indie.

Joc d'art Experimental:

La combinació dels diferents estils gràfics que conviuen dins el joc fan que se'l pugui considerar com a joc experimental. Els personatges i mapa col·lisionable utilitzen gràfics amb colors plans i senzills. En contrast, els fons de la pantalla són altament texturitzats i amb degradats. A més a més, en diversos punts del joc apareixen elements decoratius i de disseny en píxel art. Podem veure un exemple en el qual apareixen els tres casos a la (Figura 1.1).

Figura 1.1: Pantalla on podem veure els tres estils.

Un Jugador en local:

L'experiència de joc està pensada per un sol jugador. Aquest jugarà en una mateixa màquina sense requerir connexió a cap altre dispositiu ni xarxa de connectivitat.

1.4 Anàlisi del mercat

Per tal de conèixer i comparar productes similars als que m'he proposat desenvolupar, he portat a terme una recerca que ha tingut com a font principal algunes plataformes de distribució de jocs. En aquest sentit, la meva experiència prèvia com a jugador i creador de videojocs ha estat un punt de partida avantatjós alhora de saber per on començar. A continuació descriure...[]

Cercador de contingut

Com a primer pas calia consultar en cercadors de contingut, i vaig utilitzar el de Google [12]. Les paraules clau de les cerques van ser entre d'altres: `2D`, `jocs`, `videojocs`, `ganxo`, `joc de tir`, `joc de plataformes`, `un jugador`, `indie`, `midcore/hardcore`. Per fer més eficient i més àmplia la cerca vaig combinar aquestes paraules en diversos idiomes i en diferents frases. La quantiat de resultats va ser molt gran en tots els casos, però els resultats d'aquestes cerques eren jocs que compartien alguna característica amb el meu, però cap significativament similar, o directament jocs que no tenien cap interès per l'estudi.

Distribuïdors de videojocs

Com a segon pas, vaig buscar en coneguts distribuïdors online de videojocs. Vaig triar GOG [9] i STEAM [22] per la seva popularitat, i perquè engloben tant l'escenari indie com el més comercial. En aquestes dues plataformes podem trobar contingut per a diversos dispositius, però són conegudes principalment pels jocs d'ordinador. En ambdues plataformes les paraules clau de les cerques van ser similars a les descrites anteriorment, i els resultats també van ser equivalents.

Jocs de plataforma portàtil

Com a tercer i últim pas, vaig cercar en diferents tipus de plataformes portàtils, centrant-me per una banda en jocs per a consoles com PlayStationPortable, GameBoy Advance i NintendoDS, i per l'altra en jocs per a dispositius mòbils. En el primer cas vaig buscar en reculls de videojocs de les empreses de cada consola [21, 18]. En quant als dispositius mòbils, vaig utilitzar la app store de Google, la de IOs, i la d'Amazon. Els resultats no van diferir dels obtinguts amb les altres plataformes.

Conclusions

Els resultats més rellevants de la nostra recerca van ser els següents:

1. Aquells jocs que utilitzaven la mecànica del ganxo, l'usaven de forma primària i necessària pels reptes del joc. Eren especialment jocs de plataforma de tipus puzle, scrollers o runners. En general, la mecànica era utilitzada com a únic mètode per esquivar obstacles, arribar a llocs o resoldre situacions.

2. Gràficament, no vaig veure cap joc de les característiques del que jo proposava. Majoritàriament es repartien en dos grups. El primer amb imatges molt detallades, amb personatges molt ben fets i amb molts detalls decoratius. El segon amb art purament minimalista ressaltant les ombres i els canvis plans de color o els degradats.

3. Entre els jocs de tir (Shooters) en dues dimensions, en la majoria dels casos es permetia disparar només en direcció horitzontal i vertical, o en algun cas, amb no massa menys restriccions, com per exemple direccions limitades a angles múltiples de 45º. També hi havia jocs en els quals el temps no era important, i es prioritzava disparar amb precisió, normalment per tal de resoldre un puzle.

4. En quant a coincidències, cal destacar que la forma més usual d'interacció amb l'usuari era l'ús de comandament o ratolí, i rarament amb pantalles tàctils. Això pot ser degut al fet que la interfície dels mòbils no ofereix la precisió necessària en aquesta mena de jocs, i pel mateix motiu, la majoria d'aquests jocs eren per a ordinador. En general, utilitzaven físiques 2D, eren d'un sol jugador, i eren principalment indies o d'empreses petites. ...

1.5 Influències

El disseny d'un videojoc sempre està fortament condicionat per l'experiència prèvia del dissenyador. En aquest apartat comento breument alguns dels projectes que coneixia amb anterioritat i que han influït al desenvolupament del joc.

Teeworlds:

Joc de plataformes en el qual s'utilitzen armes de foc per disparar i un ganxo per ajudar al moviment. És multijugador per xarxa. Els jugadors s'enfronten entre ells (en solitari o en equips) per guanyar partides normalment de curta durada. A més és un projecte de codi obert construït per una comunitat molt entregada que dona suport als desenvolupadors del joc. Per més informació podeu visitar la seva pàgina web [3].

Figura 1.2: Pantalla de joc del Teeworlds

Thomas Was Alone:

És un joc molt senzill en el qual l'ambientació i la història narrativa cobren força, ja que els gràfics són molt plans, però genera un ambient molt peculiar i íntim. Ha estat capaç de crear seguiment i interès en gran part de la comunitat indie. El creador del joc és Mike Bithell [4].

Figura 1.3: Escena del joc Thomas was alone

Muffin Knight:

Es tracta d'un joc pseudo 3D. Els gràfics són models en tres dimensions, però la forma de jugar (Gameplay) és totalment 2D. Per progressar, s'han de derrotar enemics que van apareixent en un mapa tancat i així accedir al nivell següent. A més d'una bona experiència de jugabilitat hi ha humor i molts personatges que es van revelant a mesura que s'avança. Aquest joc és de la companyia Angry Mob Games [6].

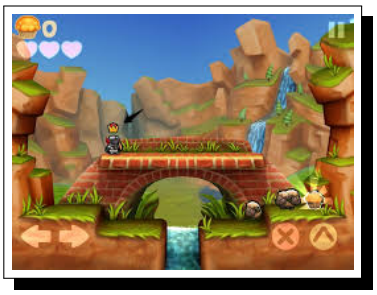

Figura 1.4: Pantalla de joc del Muffin Knight

Gavity Falls:

Finalment, m'agradaría citar aquesta sèrie de televisió. No perquè m'hagi influenciat en les mecàniques i tipus de joc, si no per l'art. Els aspectes gràfics de colors plans per els personatges principals i fons fortament texturitzats funciona de forma magistral en aquesta sèrie de la companyia Disney, produïda per Alex Hirsch amb Ian Worrel com a director d'art. [19], [5], [13].

Figura 1.5: Pantalla de joc del Muffin Knight

1.6 Tecnología i èines utilitzades

Per fer un videojoc és de vital importància triar curosament les eines a utilitzar. Dins d'aquest apartat veurem la tecnologia sobre la qual s'ha construït el codi, i el llenguatge de programació utilitzat. Explicaré les diferents opcions estudiades i els principals motius pels quals han estat triades o descartades.

Tecnologia

Actualment hi ha moltes tecnologies sobre les quals programar un joc. Per tal de reduir les opcions a estudiar per aquest projecte, la tria es va fer a partir de tecnologies conegudes. A continuació es presentaran les alternatives amb una breu descripció de concepte, i una explicació de la presa de decisions.

Alternatives:

1. Unity [23], UnrealEngine [7] o Defold [17] ofereixen la possibilitat de fer un joc en un editor amb el motor de joc, les físiques i altres components ja preparats.

2. SDL [25], o SFML [10] permeten fer un joc a més baix nivell amb funcions bàsiques, però obligant al programador a fer el seu propi motor de joc, físiques, animacions...

3. OpenGL [14] serveix per a fer un joc totalment des de la base, i utilitzant poc codi preconstruït. Tot i així, com que no dona suport a certs aspectes necessaris en un joc, com per exemple l'àudio, normalment s'utilitza juntament amb alguna llibreria externa addicional.

Presa de decisions:

Escollir OpenGL hauria condicionat menys la programació, i s'hauria pogut prescindir d'algunes dependències de codi. Tanmateix, el cost de la gestió de baix nivell del joc hauria comportat moltes més hores. Donat que el temps del projecte era limitat, la decisió es podia reduïr en fer un joc més senzill i bàsic (de cara a l'usuari) amb OpenGL, o apostar per una experiència d'usuari més rica utilitzant una base de menys baix nivell. La independència que oferia era atractiva, però al final la decisió va ser poder dedicar més temps a les funcionalitats del joc que no pas a la base. Les llibreries del segon bloc, permetien entrar en detall i treballar amb la implementació pròpia de tots els aspectes del joc en sí, sense les preocupacions de la gestió de finestres, textures, sprites... que es tracten a més baix nivell. La opció dels motors de joc del primer grup també tenia avantatges, però ja era un nivell d'abstracció massa gran, i no et donen la independència d'aquestes plataformes, ni el control específic dels diferents sistemes. En particular, fer les físiques i tractament de recursos i interacció amb el jugador ha estat una experiència molt interessant i enriquidora. Era una decisió que em comportaria més feina, però es compensaria en l'aprenentatge. D'entre SDL i SFML, vaig escollir la segona, perquè ja hi tenia experiència, i perquè coneixia tant la seva documentació, que és de gran qualitat, com l'activa comunitat que li dona suport.

LLenguatge de programació

El llenguatge de programació és probablement la decisió que afecta més al desenvolupament del projecte.

Alternatives:

SFML originalment està enfocat a C++. Tot i així, la comunitat s'ha encarregat de fer ports (traduccions) a altres llenguatges, com per exemple C, .Net, Crystal, D, Euphoria, Go, Haskell, Java, Julia, Nim, Ocaml, Pascal, Python, Ruby o Rust. [11].

Presa de decisions:

De les diferents alternatives, els llenguatges que em resultaven més propers, i amb els que per tant hi podia treballar millor, eren C++, python i Java. Per una banda Java em donava l'avantatge de fer un programa multiplataforma amb més facilitat, i python la capacitat de desenvolupar més ràpidament. Java té l'inconvenient de no permetre gestionar els recursos directament, ja que el garbage collector pren el control d'aquesta tasca. Python és un bon llenguatge d'scripting, però no convé utilitzar-lo per projectes que requereixin molta escalabilitat o de gran mida en general. De fet, no està dissenyat amb aquest objectiu i acostuma a acabar generant problemes. C++ és més robust en aquests dos aspectes, i compta amb el valor afegit de ser el llenguatge amb el que més he programat tant dins com fora de la universitat.

Així doncs, la decisió final va ser treballar sobre SFML utilitzant C++.

Èines de control de versions i de projecte

Per treballar i mantenir un registre i un control de la part de codi, he utilitzat **GitHub**[8] amb la llicència d'estudiant universitari. GitHub és un sistema de control de versions per programes. Utilitzant Git et permet controlar, treballar i modificar el teu projecte, tot mantenint-ne el registre dels canvis i l'evolució. A més a més, degut a que la informació no solament es guarda a la màquina local sinó també en un servidor extern, ofereix la possibilitat d'accedir-hi des de qualsevol lloc del món a través d'una connexió per terminal o a través d'una pàgina web. Addicionalment, manté una còpia de seguretat per si es perdés el projecte local, i per poder anar enrere en el temps si es donés el cas que s'ha de desfer algun canvi o consultar alguna implementació anterior.

Per la part de disseny, vaig treballar especialment en suport físic (paper i llibretes). Això em garantia l'accessibilitat independent de l'ordinador i la possibilitat d'escriure i fer dibuixos a mà.

En quant al desenvolupament del projecte en sí, vaig crear un **Trello**[24], que és una eina de treball àgil [26] que et permet mantenir etiquetes amb tasques a fer, i columnes que vaig anomenar *Coses a fer*, *Properes coses a fer*, *Coses que estic fent* i *Coses fetes*. Amb aquestes columnes podria seguir el procés de desenvolupament. Anteriorment, ja l'havia utilitzat en projectes compartits amb companys de la universitat, però en el projecte actual no ha resultat tant útil com llavos. Per una banda, en alguns moments no tenia internet o un navegador a mà, i les tasques quedaven desactualitzades i es feia més complex anar seguint el progrés. A més a més, en aquest projecte he estat treballant sol, així que no hi havia col·lisions de feina, ni d'idees, ni hi havia discrepàncies a resoldre, ni coses a discutir amb la resta de l'equip, i tampoc la necessitat de sincronització i

repartició de tasques. Sabia en tot moment el que hi havia fet, el que hi hauria d'haver, el que volia que es fes, i com fer-ho. Quan volia canviar alguna cosa, preferia fer-ho directament, estalviant el temps i la dedicació de mantenir la taula del Trello. És cert que a vegades he utilitzat *post-it*s per recordar alguna cosa concreta, però només com a solució puntual a problemes de poc abast.

Val a dir que Trello és una eina molt útil, però que ho és més quan treballes en equip o en més projectes simultàniament. Com que he estat completa i exclusivament centrat en aquest, m'ha acabat resultant més fàcil prescindir-ne.

Capítol 2

Descripció i Continguts del joc

2.1 Personatge principal

El personatge principal és un ser gràficament minimalista (Figura 2.1). El seu cos és una esfera, i el color és completament pla. Sobre el cos es troben els dos ulls, inexpressius i senzills, i al voltant, seguint la posició del ratolí (o de la direcció senyalada amb el comandament), una altra esfera. Al llarg del joc anirà adquirint diferents barrets que portarà al costat esquerre del cap. En cap moment parla ni mostra expressions, ajudant a crear així un ambient de misteri. Tot el que el jugador sap és que li han robat el barret mentre estava en un prat de flors, i ha entrat corrent a una cova a recuperar-lo.

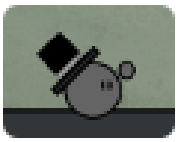

Figura 2.1: Personatge principal

Per contrarrestar la simplicitat del personatge, calia integrar-lo. Les físiques de moviment donen una sensació de realisme, ja que el moviment té fregament i acceleració. Al seu torn, el moviment del ratolí (o direcció des del comandament) no només fa moure l'esfera exterior si no també els ulls d'una forma especial. A més a més, els barrets també són afectats per les físiques del personatge. Quan es mou endavant el barret s'inclina endarrere, i endavant en el sentit contrari com il·lustra la (Figura 2.2). Al seu torn, quan el jugador salta o cau, el barret se separa del seu propietari potenciant la sensació de dinamicitat i moviment. En caure al terra, el barret tornarà a la posició original. Podem veure aquest efecte a la figura (Figura 2.3).

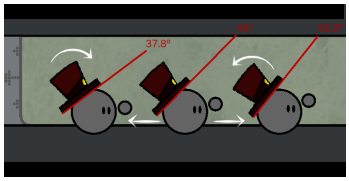

Figura 2.2: Moveiment horizontal

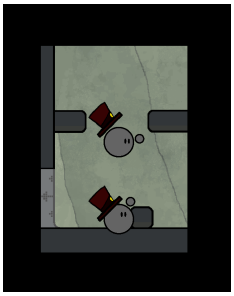

Figura 2.3: Moviment vertical

Els ulls del personatge es mouen de forma similar a la esfera exterior, però amb una trajectòria més el·líptica que dona una sensació més coherent i de volum al jugador. Una alternativa era fer moure els ulls en dues dimensions d'esquerra a dreta sense moviment en y. Aquest moviment seria 2D però no coherent quan jugador apunta enlaire. Una altre opció era que seguís una trajectòria circular com la esfera exterior. D'aquesta manera el jugador queda coherent, però perd la sensació d'esfera, passant a ser un cercle pla. A més a més, l'esfera exterior i els ulls haurien format un bloc amb el mateix moviment i generaria una falsa sensació de grup entre els dos elements independents.

Podem veure les diferents versions a la (Figura 2.4).

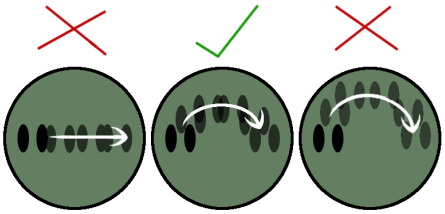

Figura 2.4: Moviment utilitzat per els ulls

2.2 Pantalles

Al llarg de l'aventura hi ha diferents pantalles cadascuna amb el seu sistema i la seva implementació darrera. A continuació podrem veure quines són.

Portada
Una imatge d'inici del joc amb una imatge representativa i el títol del joc.

Figura 2.5: Portada

Splash Screen
Una primera pantalla amb una imatge i un text. Normalment aquesta pantalla s'utilitza per mostrar una imatge corporativa de l'empresa del desenvolupament mentre es carreguen recursos del joc. [16]

Figura 2.6: Splash Screen

Menús d'Idioma i dificultat
Un menú en el qual es mostren diverses opcions en text sobre botons. En clicar un botó, la imatge canviarà per donar resposta visual al jugador de que s'ha seleccionat aquella opció. En deixar de clicar el botó l'elecció quedarà fixada i el joc procedirà a la següent pantalla.

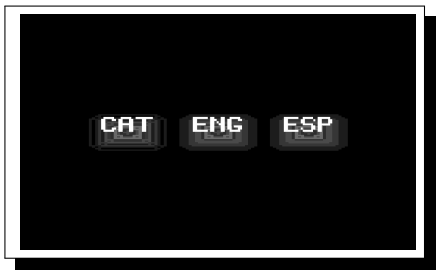

Figura 2.7: Menús

Animació inicial
Animació creada amb un sistema d'animacions propi en el qual es veu el personatge distret entre flors, quant de sobte una pinça metàl·lica s'acosta per darrera i li roba el barret. Aquest, en veure que s'emporten el seu barret, corre darrere la pinça que s'endinsa dins una cova, i hi entra al darrera. Si el jugador no vol veure la animació, en clicar qualsevol tecla la saltarà. Amb aquesta animació el jugador adquireix un context de la aventura, i una justificació per la qual el personatge està avançant per la cova dels barrets infatigablement, i acaba quan recupera el seu barret original.

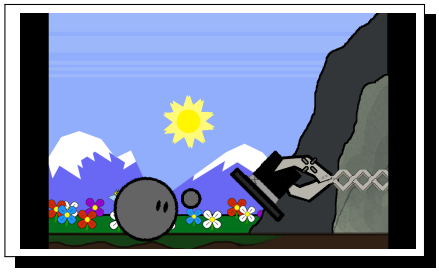

Figura 2.8: Animació Inicial

Pantalles de tutorial

Es tracta de cinc pantalles en forma de túnel en les quals el jugador ja pot començar a jugar, i va aprenent els diferents controls del joc de forma interactiva. En la primera s'explica com moure el personatge i apuntar, així que el jugador pot veure el moviment del personatge i com es posiciona segons a on apuntem. Al moure's el jugador pot portar al personatge fins a porta que condueix a la següent sala. A la segona pantalla s'explica com utilitzar el ganxo, en un túnel també obert perquè el jugador pugui provar-ne el funcionament. A la tercera pantalla es presenta la opció de saltar, i s'avisa que això només és possible en la modalitat normal (en el mode difícil no es pot saltar). Per obligar al jugador a aplicar el concepte, es bloqueja el camí amb un bloc que s'ha de saltar (o esquivar amb el ganxo). A la quarta pantalla podem veure una petita explicació en la qual es presenta el concepte de vides i de temps, perquè el jugador tingui un context quan vegi els elements a la pantalla de joc. En la cinquena i última, s'explica com entrar als nivells. El disseny de les pantalles està fet per poder-ho veure sense necessitat del tutorial, però per precaució he decidit donar la doble informació.

Figura 2.9: Pantalles de Tutorial

Pantalles de nivell

Cinc pantalles en les que es pot entrar als corresponents tres nivells que s'han d'anar superant. Veurem més concretament com funcionen en el següent apartat.

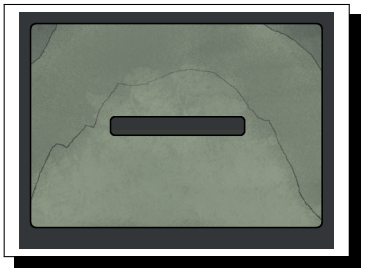

Figura 2.10: Pantalla de Nivell

Pantalles de túnel
Similars als tutorials, són túnels a través dels quals el jugador ha de passar per anar a la següent cova. En aquests hi ha elements decoratius i text, però s'utilitza només per fer referències a altres jocs, i per donar més contingut a l'experiència.

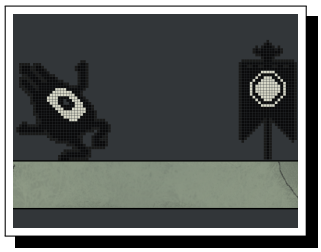

Figura 2.11: Pantalles de Túnel

Pantalla de crèdits
Mostra els crèdits del joc, presentant una imatge de final, el concepte del qual donar crèdit, i els autors de les diferents parts, en les que s'inclou programació, game design, gràfics, sons i música.

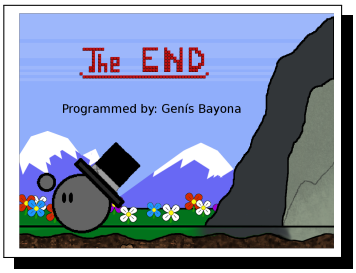

Figura 2.12: Pantalla de Crèdits

2.3 Enemics

Al llarg del joc, a mesura que anem desbloquejant nivells, aniran apareixent també nous enemics. A continuació veurem tots els que ens podem trobar.

- **Bola de punxes** (Figura 2.13)
  Enemic que va rodant pel mapa canviant de direcció quan xoca amb una paret. Són els primers enemics que apareixen, són fàcils de guanyar, però poden causar problemes si s'acumulen molts.

Figura 2.13: Bola de punxes

- **Bola de punxes saltadora** (Figura 2.13)
  Enemic que es desplaça pel mapa saltant. És gràficament igual que la Bola de punxes presentada anteriorment, però en comptes de rodar va saltant pel mapa. Quan salta, l'enemic es deforma per donar més coherència a l'acció. Són perillosos perquè poden tocar al jugador mentre salta o està enganxat amb el ganxo. A més a més, dificulta preveure qui salta i qui roda quan hi ha múltiples enemics.

- **Bloc** (Figura 2.14)
  Enemic format per un bloc massís. És molt gran i ocupa gran part de la pantalla. El seu comportament de moviment consisteix en moure's pel nivell enganxat a les parets, amb certa acceleració. Requereix més bales que la resta per guanyar-lo, i la seva mida pot ser un problema en nivells amb molts enemics.

Figura 2.14: Enemic Bloc

- **Estrella** (Figura 2.15)
  Enemic en forma d'estrella que va rebotant pel mapa en linia recta i direccions aleatòries. Cada vegada que xoca amb un obstacle del nivell, ja siguin les parets que el limiten o una plataforma interna, canviarà de direcció. Són enemics imprevisibles i que es poden moure per tot el mapa, però degut a que necessiten menys cops per desaparèixer són també enemics fàcils de derrotar.

Figura 2.15: Enemic Estrella

- **Fantasma** (Figura 2.16)
  Enemic semitransparent que es mou pel mapa ignorant les parets. El fet que siguin grans i que puguin travessar els obstacles del mapa fa que un bon refugi deixi de ser-ho en qüestió de segons. A través del seu cos es pot veure el fons gràcies a que aquest és semitransparent. Els fantasmes són les úniques entitats del joc que mostren aquesta semi-transparència. Com les estrelles, el seu moviment és recte en una direcció, i canvien de direcció quan xoquen amb una paret. Però en aquest cas només ho fan amb les parets que delimiten el nivell.

Figura 2.16: Enemic fantasma

- **Ninja** (Figura 2.17)
  Enemic de petites proporcions que es mou ràpidament pel mapa enganxat a les parets, però saltant de tant en tant en alguna altra direcció. Aquests enemics són complicats de disparar perquè són bastant més petits que la resta, però a més a més, quan reben un impacte, poden ignorar-lo, fent aparèixer un tronc al seu lloc (Figura 2.18), que caurà al mapa durant uns segons, i que valdrà més no tocar si el jugador vol conservar les seves vides.

Figura 2.17: Enemic ninja

Figura 2.18: Tronc de protecció del ninja

- **Disparador** (Figura 2.19)
  Enemic que es mou per terra i que canvia de direcció quan xoca amb una paret. Cada cert temps dispara un projectil contra el personatge. Aquests projectils poden ser destruïts si el jugador dispara contra ells.

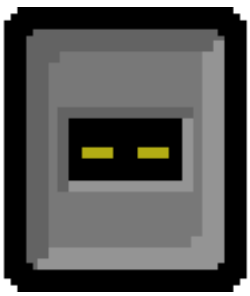

Figura 2.19: Enemic disparador

- **Enemic final** (Figura 2.20)
  L'enemic final és un barret de copa gegant similar al que el personatge ha entrat a buscar a la cova. Aquest enemic té molta vida, i certes peculiaritats. El moviment és similar al dels ninjes, però quan el jugador el dispara passen dues coses. Per una banda, que decreix en mida, fent-se més petit, i augmentant la velocitat. Per l'altra, fa aparèixer un enemic aleatori al mapa al seu costat. Aquesta mecànica és molt interessant, perquè obliga al jugador a anar eliminant els enemics de mica en mica, i aquells que volen anar massa ràpid i atacar-lo moltes vegades, ràpidament queden atrapats en un mar d'enemics dels que no poden escapar. Al seu torn, però, degut a la seva mida, obliga a haver-lo de atacar ràpidament per tal de poder esquivar-lo més fàcilment. D'aquesta manera, el jugador ha de trobar la forma d'atacar ràpid però alhora acabar amb els altres enemics que fa aparèixer ràpidament, i simultàniament mirar de no perdre les vides. A mesura que avança el nivell, l'enemic final és cada vegada més petit, i per tant més fàcil d'esquivar i d'eliminar als altres enemics del mapa, però alhora més difícil de tocar. Per tant requereix més precisió en disparar, i obliga el jugador a no perdre el temps que es va esgotant lentament.

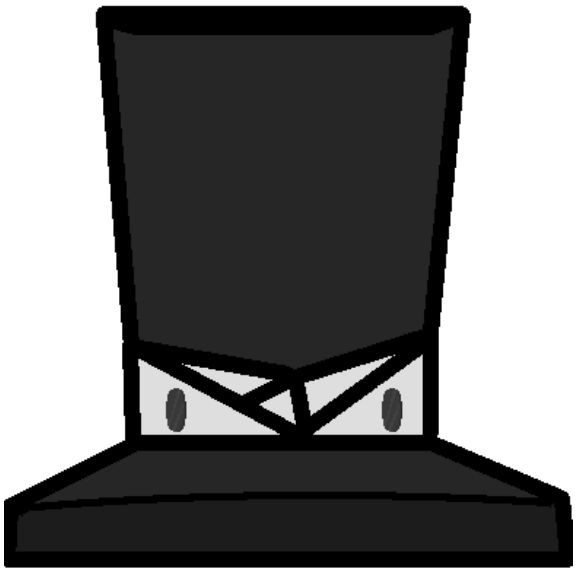

Figura 2.20: Enemic final del joc

2.4 Nivells

Els nivells es poden separar en coves d'elecció de pantalla, i pantalles de joc. En total hi ha cinc coves, cada una d'elles amb tres barrets. Cada barret ens portarà a una pantalla per superar. Una cova inicialment té un barret desbloquejat, dos interrogants on tindrem els dos altres barrets, i la porta oposada a la que s'ha entrat tancada. Els barrets desbloquejats van rotant perquè no es confonguin amb decoració. Per entrar al nivell d'un barret concret (que tinguis desbloquejat) s'ha de disparar al mateix. Això s'explica al tutorial. Quan el jugador entra a una cova nova, només hi ha un barret, però quan n'hi ha més d'un desbloquejat, pot haver-hi algun problema. Quan volem entrar a un nivell concret, podem intentar disparar un barret i tocar el del costat sense voler. Per prevenir aquest cas, i per ratificar que el jugador vol entrar a un nivell concret, quan dispari a un barret, aquest creixerà en mida. Tot i fer-se més gran (fent així que sigui més fàcil tornar a disparar-lo), no arrencarà el nivell corresponent fins que es dispari per segona vegada.

Una vegada entrem a un nivell, començaran a aparèixer enemics. Per poder desbloquejar el següent barret o passar a la següent cova caldrà derrotar-los a tots abans que s'acabi el temps o el personatge es quedi sense vides.

Dins la pantalla del nivell podem veure dos components visuals que no eren visibles durant la selecció. A la part superior de la cova, apareixerà una barra que indicarà el temps restant per completar el nivell. A mesura que el temps es vagi reduint la barra es farà més petita, i canviarà de color a un vermell més fort per cridar l'atenció del jugador posant així un factor de pressió. A la part inferior de la cova apareixeran les vides del personatge. Es representaran en forma de cors, i cada vegada que rebi un impacte, un d'aquests cors desapareixerà fent una animació en la qual la mida es va reduint fins que desapareix. Quan hi hagi un impacte que baixi la vida del jugador també es notificarà de forma visual fent que el barret pampalluguegi en vermell. Durant aquest temps de pampallugues el personatge serà invulnerable.

Si el temps del nivell s'esgota o s'acaben les vides del jugador, aquest perdrà la partida i tornarà a la cova amb els barrets tal com es trobava al entrar al nivell. Des d'aquí podrà triar si tornar al mateix nivell o anar a un d'anterior per practicar.

Si el jugador aconsegueix derrotar a tots els enemics del nivell amb el temps donat, també tornarà a la cova d'origen. Si hi havia barrets per desbloquejar, tindrà el següent desbloquejat. Si estava jugant l'últim barret de la cova, es trobarà amb la porta de la dreta oberta per poder seguir a la cova següent.

Cada cova amb els seus tres nivells té el seu propi mapa, a continuació veurem una mica de disseny d'aquests.

1. El primer és un mapa senzill, amb una única plataforma que actua discretament de protecció del jugador. Es tracta de la primera cova, i ens interessa que el jugador pugui superar fàcilment els primers nivells. Per aquest motiu, aquesta única plataforma està posada estratègicament per mostrar el moviment dels enemics mentre el jugador està refugiat a sota. Una vegada vist el comportament, l'enemic arribarà al nivell del jugador (la part baixa del mapa) en una punta del mapa, anant en direcció contrària. De seguida, xocarà amb la paret, mostrant que canvia la direcció. Mentrestant, el jugador pot anar-lo disparant tranquil·lament, fent que quan l'enemic estigui molt pròxim desaparegui i no arribi a fer perdre vides al jugador al xocar-hi. A més a més, la plataforma cobreix el cas en que el jugador no ha estat pro ràpid per eliminar els enemics i s'aparta o fins i tot que l'hagin arribat a tocar. El jugador pot esquivar els adversaris saltant o amb el ganxo, però si es col·loca sobre la plataforma una vegada hagin aparegut tots els enemics, ningú el podrà tocar, i tindrà temps de pensar la millor estratègia a seguir per derrotar-los còmodament des d'una posició estratègica.

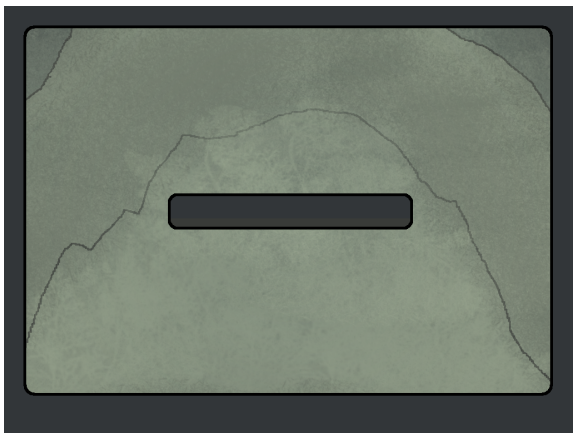

Figura 2.21: Primer mapa

2. El segon mapa té moltes parets petites. Està pensat perquè el jugador vegi les conseqüències dels obstacles al mapa, com es mouen els enemics rebotant en ells, i a continuació com els fantasmes hi passen a través. Aquest mapa fa que el jugador hagi de moure's contínuament trobar per on atacar als enemics. També per incentivar el moviment pel mapa, cal evitar els fantasmes que ignoren les parets. Aquest és un mapa de aprenentatge de moviment i de funcionament dels nous enemics.

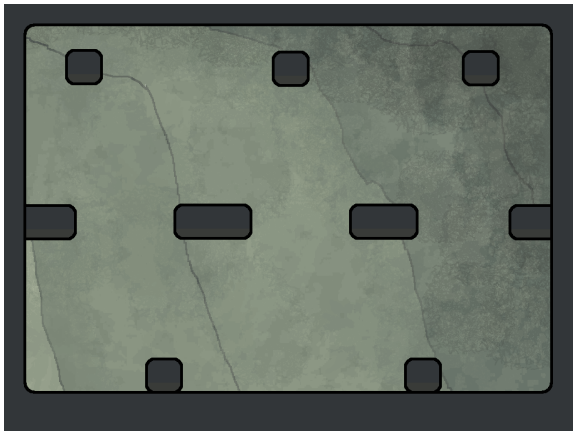

Figura 2.22: Segon mapa

3. El tercer mapa, al seu torn, té tres plataformes posades estratègicament per la presentació dels ninges i els blocs. La distribució fa difícil al jugador poder atacar-los, i el posa en perill al intentar-ho, ja que els enemics van donant voltes al voltant de la plataforma superior. Degut a això, el jugador ha de preveure els moviments dels diferents enemics per poder superar el nivell, que a més té el temps reduït, i obliga a anar molt ràpid.

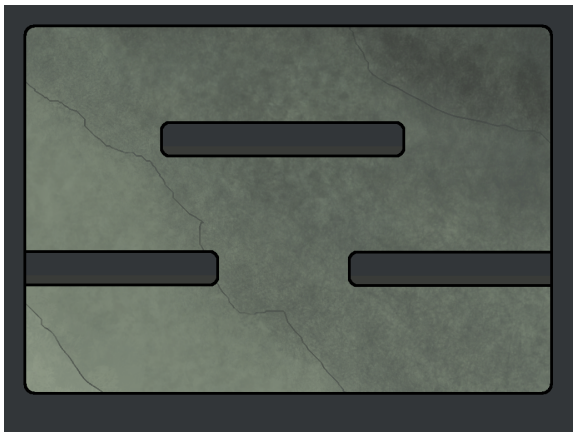

Figura 2.23: Tercer mapa

4. El quart mapa, està pensat per aprendre a jugar des de l'aire. Té dues plataformes als costats, que permeten al jugador parar i refugiar-se dels enemics que apareixen a aquest nivell que disparen contra el personatge. La idea és que el jugador pugui refugiar-se allà i sortir amb un salt o el ganxo, mentre dispara, atacant als enemics que es van agrupant a la part inferior del nivell, i tornar a un dels dos costats per seguir-se protegint. Aquí el jugador ha de aprendre el joc aeri i conèixer els nous enemics.

Figura 2.24: Quart mapa

5. El cinquè i darrer mapa, és completament obert, sense obstacles. Això és així perquè és el nivell en el qual hi ha de haver l'enemic final que amb no es podria moure degut a que és molt gran i si hi hagués obstacles quedaria atrapat. A més el jugador necessitarà tot l'espai per poder moure's i esquivar els enemics. Abans de l'enemic final, dins d'aquesta cova, encara hi ha una pantalla en la qual es forçarà a usar el ganxo al jugador, perquè quan jugui contra l'enemic final ja tingui totes les habilitats ben apreses.

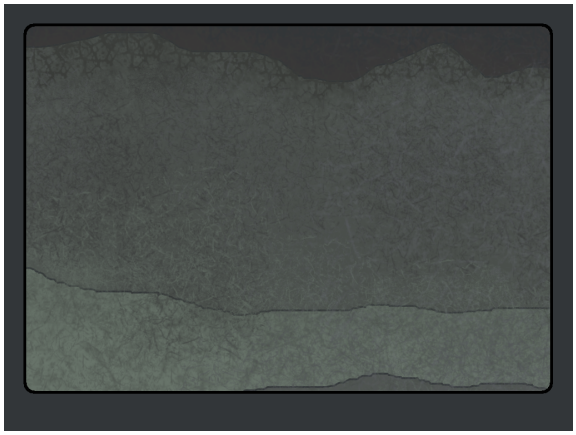

Figura 2.25: Cinquè mapa

2.5 Idiomes disponibles

Els textos del joc estan disponibles en tres idiomes: Català, Anglès i Castellà. A l'inici del joc, abans de trobar-te amb altres textos es demanarà al jugador l'idioma que prefereix a través del menú d'idiomes. Aquest idioma es guardarà per la resta del joc, i s'utilitzarà en els diferents textos del joc com les opcions de dificultat, els tutorials i els textos d'ambientació.

Figura 2.26: Sel·lecció d'idiomes

2.6 Dificultats

A l'inici del joc es pot triar la dificultat amb la que vols jugar. Es pot triar entre normal i difícil. En la modalitat difícil, el joc serà el mateix amb els mateixos nivells, enemics... però les pantalles seran més complicades de superar. Per una banda es perd la possibilitat de saltar, obligant a utilitzar i perfeccionar l'ús del ganxo com a medi de moviment. Per l'altre, el jugador tindrà menys vides que en la modalitat normal passant de 5 a 3. El mode difícil està pensat per jugadors experimentats, ja sigui per jugar freqüentment jocs d'aquest estil com perquè han superat el joc en mode normal i volen tornar-hi amb un repte més difícil. A més a més, la modalitat normal és una forma de permetre jugar al joc sense requerir l'ús molt continuat del ganxo, que és una mecànica complicada, però per adquirir la completesa, forçar a haver-lo de dominar.

Figura 2.27: Sel·lecció de dificultat

Capítol 3

Disseny i Programació del joc

3.1 Sistema d'escenes

Al llarg del joc s'utilitzen diferents escenes. Cada una té les seves peculiaritats i necessitats individuals, però també moltes similituds, com a mínim estructurals.

Necessitava que totes elles s'inicialitzessin, donar-los el control, i finalment esborrar-les. A més a més, també necessitava una forma de moure'm d'una a la següent que fos versàtil, ja que podia haver de saltar entre elles de diferents formes, i especialment al iniciar el joc, tenint en compte els possibles canvis calia fer un sistema genèric.

Per resoldre aquest problema, vaig crear una classe genèrica d'escena de la qual heretarien les altres. Aquesta tenia funcions i atributs que podien resultar útils a les diferents classes, i unes concretes que em permetien treballar sobre elles amb un gran nivell D'abstracció, sense importar qui n'heretés i les peculiaritats concretes. Les funcions concretes per a aconseguir aquest nivell d'abstracció eren senzillament una funció `init`, que serveix perquè l'escena s'inicialitzi amb els valors que necessiti en el moment d'entrar en acció i la funció `run` amb la qual l'escena agafarà control del fil d'execució.

Amb aquestes funcions, podem posar totes les escenes en una estructura, i canviar l'escena que controla el joc en qualsevol moment. Per fer això executarem la funció `init` de l'escena desitjada, i li cridem la funció `run` per donar-li el control de la execució. Aquest procés es farà des de la classe principal del joc. L'estructura utilitzada per guardar les diferents escenes és un std::map<std::string,Scene*> . Al arrancar el joc es creen les escenes i es guarden al mapa amb un nom associat. Dins les escenes mateixes, podrem cridar una funció de la classe principal anomenada `changeScene`, que rep com a paràmetre un string que representa el nom de l'escena a la que volem canviar. Amb aquesta funció, la classe principal sap que cal carregar una escena nova, així que cridarà la funció `init` i `run` com hem vist abans, i donarà el control a l'escena desitjada, associada amb el nom passat com a paràmetre de la crida a `changeScene`. Tenir noms associats a les escenes és molt útil perquè ens permet navegar de forma senzilla entre elles, i ens permet utilitzar comandes com a noms genèrics. Per exemple en el meu sistema si utilitzes el nom *previous* en la comanda `changeScene`, el sistema torna a l'escena en la que et trobaves abans de saltar a l'actual.

Classe Scene

La idea era que cada subclasse d'Scene pogués implementar-se els seus propis mètodes i comportaments, però compartir a part de les funcions ja comentades, una estructura de bucle de joc (game loop) [20], que no calia duplicar a cada una. Per això, la funció `run` de l'Scene implementa aquest game loop. En ell es processa l'entrada (input) que es fa en un sistema independent, es controla el temps i el frame rate (el temps dedicat a cada iteració del bucle de joc) per mantenir-nos a 60 frames per segon (iteracions del bucle per segon), i s'executa la funció `update` i `display` . La primera haurà de ser sobreescrita per cada Escena filla que vulgui tenir un comportament depenent del temps, la segona és una mica més complicada, ja que s'encarrega d'esborrar la finestra, posar la view (vista de la càmera) de l'escena, cridar la funció `render` on cada subclasse podrà posar tot allò que calgui renderitzar a la pantalla, i finalment restituïr la view per defecte, i pintar el punter del ratolí. Aquest últim pas es fa així perquè el punter es pinti al punt que volem de la pantalla independentment de la càmera i sistema de coordenades utilitzat dins de cada escena.

3.2 Controlador de recursos (Resource Manager)

El joc desenvolupat utilitza alguns recursos, i per treballar amb ells he creat també un sistema especialitzat. Un dels principals problemes que comporta l'ús de diferents recursos al llarg del joc és el fet de haver de repartir codi de càrrega en diferents punts del programa, i la repetició d'alguns d'aquests recursos que s'utilitzen en diferents llocs. Aquest sistema funciona per qualsevol recurs que vulguem guardar-hi, però en el projecte l'he utilitzat per les textures, les animacions (vectors de textures), shaders i fonts. El funcionament d'aquest Resource Manager és el següent: Primerament declarem tots els objectes que volem tenir de forma estàtica al header. Això generarà una llista bastant gran d'elements, però podrem tenir-los tots en un mateix lloc, i sempre sabrem on anar a buscar-los si s'ha de realitzar algun canvi. A continuació, carregarem cada un d'aquests elements utilitzant el path (adreça dins el sistema) desitjat i les funcions pròpies de cada tipus. Això ens permet tenir una referència de com carregar els objectes comuna a tots ells, i podem veure si hem comès algun error fàcilment. Per assegurar el fàcil descobriment dels errors, també s'escriu un missatge d'error en cas de no aconseguir carregar el recurs.

D'aquesta manera, si des de qualsevol punt del codi es vol accedir a un o més d'aquests recursos, només caldrà incloure la classe `Resources.h` , i podrà accedir al recurs desitjat utilitzant la sintaxi `Resources::recursDesitjat`. Aquest sistema no només facilita l'accés als recursos, podem veure que també ens permet utilitzar recursos de forma compartida, ja que múltiples elements poden adquirir un mateix recurs sense problema. En cas de necessitar un recurs que hagi de ser modificat per qui el vulgui fer servir, és recomanable carregar un recurs nou especificant-ne l'ús, per no tenir conflictes de modificacions entre diferents actors d'un objecte concret. Si s'ha de utilitzar en molts llocs modificant-lo, també es pot carregar el recurs a l'entorn de qui l'hagi de fer servir com faríem sense aquest sistema, ja que no es pot utilitzar un sol element general per tots si necessita ser modificat.

3.3 Controlador de sons (Sound Manager)

Els sons i música també són recursos, però he fet aquesta classe independent del Resource Manager explicat a l'aparta anterior, perquè com veurem, aquest sistema ens ofereix propietats diferents. Per una banda he generat un sistema de càrrega de música (Del disc dur a la memòria del programa) en el qual cada cançó es carrega amb un nom associat a cada pantalla. Per aconseguir això, i de forma que es pogués modificar sense tocar el codi, he utilitza un sistema de directoris amb els noms de les pantalles i subpantalles. Aquests directoris contenen el fitxer de música (en format ogg) que hi ha d'anar associat. Si trobem un fitxer anomenat *follow* el programa seguirà amb la música que tenia carregada prèviament. Si hi ha un fitxer anomenat *none*, deixarà de sonar música en aquella pantalla. Altrament, la música serà carregada, i reproduïda al arribar a la pantalla que té el mateix nom que la carpeta en la que hem posat la cançó. En cas que volguem parar la música en qualsevol moment del programa podrem fer-ho també utilitzant el nom clau *none*.

Els sons es carreguen de forma similar als recursos del Resource manager, però una vegada carregats els associem a nom concret com hem vist que es fa amb les escenes. Com que en aquest cas els associem un nom (ho fem posant-los en un map), no cal tenir un nom de variable per cada un, ja que hi accedirem a través de la paraula associada al map, i podem utilitzar un vector en el qual els tinguem tots integrats.

Una vegada tenim carregats tots els recursos, podem seguir per veure com es justifica tot aquest sistema independent de la resta de recursos. Amb aquest Sound Manager, només ens caldrà incloure'l, i des de qualsevol punt del codi podem fer sonar qualsevol só, o arrancar qualsevol música. Per cridar les funcions els passarem per paràmetre senzillament el nom que hem associat al recurs deistjat, que pot ser tant descriptiu com vulguem i independent del nom del fitxer, i el sistema s'encarregarà de reproduir-los.

Per mantenir el control, sempre accedint a través dels noms associats, podrem utilitzar les funcions següents:

```
static void playSound(std::string name);
static void playMusic(std::string name);
static void stopMusic(std::string name);
static void pauseMusic(std::string name);
static void setLoop(bool loop, std::string name);
static void setPitch(float pitch, std::string name);
static void setVolume(float volume, std::string name);
static void setPosition(float x, float y, float z, std::string name);
static void setGlobalSoundVolumen(float volume);
static void setGlobalMusicVolumen(float volume);
```

Interfície de funcions que ofereix el Sound Manager

3.4 Controlador d'entrada (Input Manager)

Aquest sistema serveix per tractar de forma única el control d'input o entrada de l'usuari cap al joc. Per una banda abstreu codi de diversos punts del joc, i permet fer el tractament de forma independent en aquesta classe. A més, gràcies a això, ens permet també associar accions a diverses entrades. Dins el codi es requerirà si una acció determinada ha estat realitzada, i el sistema li respondrà si ho ha estat o no. Per definir aquestes accions, el sistema acceptarà bindings (lligams) entre una forma d'input i una acció. Per fer-ho, es defineix un `enum` amb els noms de les diferents accions, i al iniciar el joc es fa un binding de les accions que ens interessen a una tecla concreta del teclat, un botó del ratolí, o algun component del comandament. Això també acceptaria accions artificials creades per el sistema, que podriem invocar des del codi, però per el projecte no ho he necessitat. El sistema s'encarregarà de comprovar si s'han pulsat les tecles o botons que tenen accions associats, i el codi que vulgui saber-ne l'estat només haurà de consultar a la classe `InputManager` que com els altres sistemes és estàtica utilitzant la funció `action` amb un paràmetre que indicarà quina acció volem consultar si ha estat realitzada. Algunes accions que ho requereixin podran retornar un valor numèric com per exemple el moviment del joystick del comandament que retorna la quantitat de moviment que ha realitzat.

Internament el sistema utilitza mapes que enllacen cada acció amb unes estructures especials que per cada sistema suportat (teclat, ratolí i comandament) ens indica si la acció s'ha realitzat o no. Per obtenir l'informació de si s'ha realitzat l'acció, es realitza una observació activa. Per cada element al que s'hagi associat una acció, a cada iteració del game loop, comprovarà si ha estat activat, i actualitzarà l'estat.

A continuació podem veure un exemple de com es fa el binding, on es pot veure que és intuïtiu i resulta fàcil de treballar-hi.

```
InputManager::bind(InputAction::up,    sf::Keyboard::W);
InputManager::bind(InputAction::left,  sf::Keyboard::A);
InputManager::bind(InputAction::down,  sf::Keyboard::S);
InputManager::bind(InputAction::right, sf::Keyboard::D);

if( InputManager::action(InputAction::down) ) moveDown();
```

Binding de les accions de moviment a les tecles W, A, S i D.

3.5 Sistema d'Animacions

Per l'inici del joc volia fer una petita animació d'introducció per ajudar a contextualitzar el joc. Hauria pogut utilitzar algun sistema existent d'animació, però vaig optar per desenvolupar-ne un de propi. La idea principal era poder definir una animació en un fitxer de text, i que el programa el convertís en la animació en sí.

La meva aproximació per descriure l'animació va ser la següent: Primer es descriurien els objectes que han de aparèixer a l'escena, i a continuació les diferents accions. Per fer més fàcil el control, també he implementat una forma de poder posar comentaris dins les definicions, que seran ignorades pel pàrser (o traductor de text a animació). Per descriure els objectes, primer s'ha de donar el tipus, a continuació el nom amb el qual ens hi referirem, el path on trobar el recurs, i si comença actiu o no amb les paraules claus `visible` i `hide`. Els tipus podrien ser ampliats, però per ara només suporta Sprites, ja que és tot el que em feia falta per la meva animació.

Amb els objectes ja carregats, passem a les accions. Aquí el principal repte consistia en fer un sistema que pogués realitzar diferents accions sobre els objectes de forma simultània en cada un d'ells i en diferents moments. Per fer això havia de forçar que cada objecte s'actualitzés per separat en comptes de realitzar les ordres seqüencialment. El format de les ordres és: nom de l'ordre, objecte que ha de realitzar-la, paràmetres que depenen de l'ordre (per exemple el moviment té `posició_final_X`, `posició_final_Y`), i el temps que es vol que tardi a fer-se. El programa llegeix les accions i les guarda en una cua. A partir d'aquí, comença a executar-se. El sistema agafa les accions de la cua i les associa a l'objecte especificats que l'ha de realitzar. Això ho fa continuadament per totes les accions que troba fins que arriba a una instrucció `wait`. Aquesta instrucció no és per els objectes de l'escena si no per el sistema, que esperarà el temps indicat per a prosseguir afegint accions. Mentre està esperant, els altres objectes s'aniran actualitzant. En acabar l'espera, el sistema seguirà executant les accions que quedin per tractar.

El sistema d'accions és fàcilment ampliable, però per ara compta amb les accions `move` que serveix per moure un objecte a una posició final en un determinat temps. `show` que canvia la visibilitat de l'objecte a visible. `hide` que canvia la visibilitat de l'objecte a no visible. `wait` que provoca una espera en la resolució de noves accions. `rotate` que aplica una rotació sobre l'objecte en un determinat temps. `scale` que permet escalar un objecte en un determinat temps. `fade` que permet fer desaparèixer o aparèixer un objecte passant d'opac a transparent o viceversa en un determinat temps. I finalment `die` que destrueix l'objecte. Un objecte destruït ja no es podrà tornar a fer servir, i serà esborrat de la memòria. Això ens permet tenir el sistema creant i destruint objectes tant temps com es vulgui.

A continuació podem veure un sistema bàsic de fitxer d'animació en el qual es defineix un enemic. el mou a diferents punts a diferents velocitats, espera un temps, i el torna a moure. Després d'això, per alliberar la memòria, li aplica la funció die.

```
########
# Intro #
########
#This is a comment

## Objects ##
Sprite
```

```
enemy
enemy.png
visible

## Actions ##
move
enemy
50
50
2

move
enemy
100
100
1

move
enemy
200
410
3

#The system will wait 7 seconds
#6 where the enemy will be moving
#plus one extra to see the time difference

wait
7

move
enemy
0
0
2

die
enemy
2

#The dolar simbol will anounce the end of the animation
\listing
```

3.6 Menús

Per tal de dotar d'una experiència més personalitzada al jugador i més adaptable a la seva situació concreta, he inclòs dos menús a partir d'els quals es pot escollir l'idioma amb el que es vol jugar i la dificultat de joc. Per fer aquests menús, he construït un sistema genèric amb el que resoldre aquests dos casos i d'altres que poguessin sorgir en un futur. Per fer aquests menús independents del codi del joc en sí, el que s'ha fet és convertir-los en una escena que agafa el control de l'execució i tracta la decisió independentment. Dins aquesta escena es generarà un `gameLoop` propi que serà l'encarregat de actualitzar i pintar els elements que correspongui. Per poder agafar el control del joc i permetre la tria, el menú s'executarà rebent com a paràmetres la finestra sobre la que s'està treballant, i un vector de cadenes de text (strings) amb les quals generarà els botons. Els botons seran un altre element que he creat. Consisteixen en un `sprite` rectangular amb dues imatges i un text. Les dues imatges s'alternaran en funció de si el botó ha estat clicat o no per tal de donar més informació al jugador sobre l'estat de l'acció. Un vegada escollida una de les accions, el menú retornarà com a valor a la crida que l'ha generat el nombre que ocupava a la llista de `strings` el valor seleccionat per l'usuari.

3.7 Mapes i Nivells

Els diferents mapes i nivells funcionen a partir de fitxers de configuració. Aquests s'interpreten a partir d'un codi especial similar al de l'animació, amb la possibilitat d'utilitzar els mateixos comentaris darrera d'asterisc, el símbol del dolar com a símbol final del document i la possibilitat de contenir salts de línia per separar el text.

Els mapes (tunels de tutorial, tunels d'entre nivells i mapes dels nivells) funcionen de la forma següent:

1. Nom de la imatge a utilitzar com a fons.
2. Paraula clau "doors".
3. Caràcters 00, 0X i XX indicant si les portes del mapa estan obertes o tancades.
4. Paraula clau "boundaries"
5. Descripcions de les caixes de col·lisió del mapa.
6. Símbol de dolar que representa el final del fitxer.

A continuació podem veure'n un exemple.

```
cave1.png

#OO, OX o XX
OO

#Caixes de colisio separades per "next"
#descrites com "left, top, width, height"
boundaries
next
#paret esquerre
0
0
40
770
next
#sostre
0
0
1024
40
next
#terra
0
702
1024
100
next
#paret dreta
```

```
984
0
40
770
next
#caixa de colisio central
300
342
430
58
end
\$ End Of Description Files
```

En el cas dels nivells, es presenten de la següent manera:

1. Nom del mapa a utilitzar.

2. Nombre de subnivells superats d'aquest nivell (0, 1 o 2) segons els barrets aconseguits.

3. Noms dels fitxers dels tres barrets que apareixen al nivell sent primer l'esquerra, seguidament el del centre i finalment el dret.

4. Paraula clau "timer"seguida del temps màxim de joc per cada un dels barrets en el mateix ordre que el punt anterior.

5. Paraula clau "spawn"seguida de la posició en X i la posició en Y del punt des d'on apareixeran els enemics.

6. Paraula clau "enemies"seguida de les descripcions de cada un dels subnivells, que serà de la forma següent: nivell[0,1,2], caràcters que equivalen als diferents enemics, seguit de nombres que ens indiquen el temps que ha de transcórrer fins que aparegui l'enemic corresponent en posició de la línia immediatament superior.

Els enemics que apareixen en les descripcions dels nivells corresponen als caràcters un a un de forma inequívoca, amb les següents equivalències:
Bola de punxes Rodadora 's': // Spike ball
Bloc 'w': // Wall square
Estrella 'f': // Flying star
Bola de punxes Saltadora 'b': // Bouncing spike
Fantasma 'g': // ghost
Ninja 'n': // ninja
Disparador 'd': //distant shooters
Enemic final 'm': //monster final boss
A continuació en podem veure un exemple:

```
cave1
0
hat1.png
hat2.png
hat3.png
timer
100
50
50
spawn
450
170
enemies
0
s s s
5 5 1
1
s b s b
2 2 1 1
2
s b s b s b s
2 2 2 1 1 2 1
\$ END
```

3.8 Sistema d'enemics

Els enemics funcionen de forma genèrica amb una estructura semblant a la de les escenes. Tota la lògica utilitzada es fa sobre un element Enemic genèric. Tota la resta d'enemics hereten de l'enemic genèric, i s'implementen els mètodes propis que difereixen de l'original. D'aquesta manera, si la lògica del joc aplica la funció de moviment sobre els elements dels contenidors d'enemics genèrics en els quals tindrem tots els adversaris del nivell, aconseguirà que cadascun es mogui de la forma concreta al tipus d'enemic que sigui sense haver de cridar funcions diferents per cada un d'ells, o discriminar el tipus cada vegada que calgui realitzar l'acció. Aquest sistema resulta molt pràctic per poder afegir enemics fàcilment, ja que per una banda estalvia molt de codi duplicat dels enemics que tenen funcionalitats en comú (per exemple el mateix tipus de moviment, les mateixes vides, la mateixa imatge ...) i per l'altre ens permet afegir i treure enemics fàcilment sense haver de tocar diferents parts del codi, encapsulant tot el comportament d'un tipus concret d'enemic en la seva pròpia classe.

3.9 Hook (Ganxo)

Per treballar dibuixar el ganxo, s'utilitza una sola imatge (Figura 3.1) que es va repetint al llarg de tota la corda del ganxo. Essencialment consisteix en un punt d'origen i un de destí, entre els que es mostra de forma repetida aquesta imatge, amb una esfera al final per mostrar el punt en el qual s'ha enganxat el ganxo, i que surt sempre des de l'interior del personatge (es pinta abans que aquest). Respecte a l'efecte en el moviment, s'aplica una força de tracció cap al punt de destí proporcional a distancia del punt d'inici. Això fa que si el personatge es penja del ganxo faci l'efecte de molla.

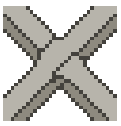

Figura 3.1: Peça de les que es composa el ganxos

3.10 Mode de Debug

Per tal de poder buscar millor els errors del codi relacionats amb col·lisions vaig implementar un mode de debug que es pot activar canviant una variable del codi. Per poder apreciar més fàcilment les col·lisions, ja que alguns elements són transparents, o les caixes de colisió no concorden perfectement amb els gràfics, s'han ressaltat les caixes contenidores o bounding boxes de cada element col·lisionable del mapa en diferents colors, per sobre dels gràfics originals.

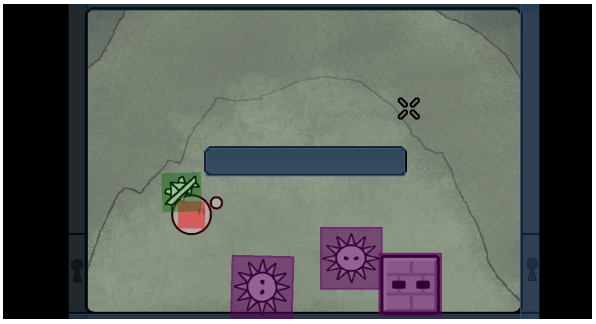

Figura 3.2: Pantalla de debug exemple 1

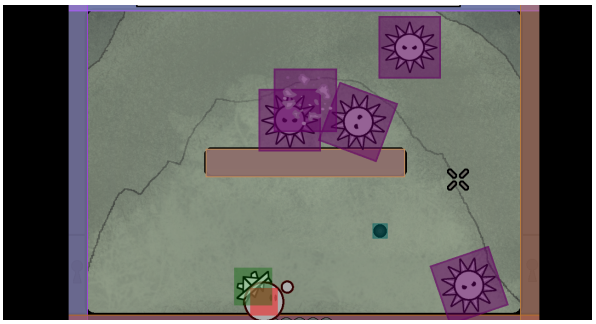

Figura 3.3: pantalla de debug exemple 2

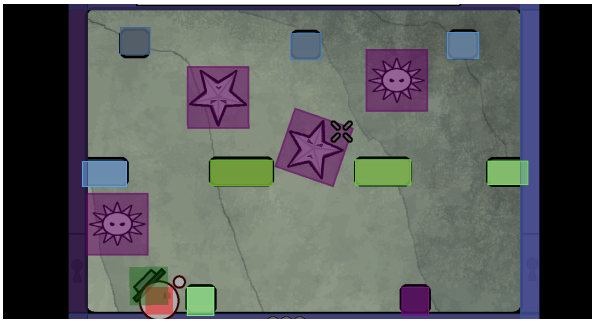

Figura 3.4: pantalla de debug exemple 3

3.11 Sistema templetitzat de logs

Per tal de fer més senzilla la localització i posterior solució d'errors i l'escriptura d'errors en cas de problemes, per exemple, quan hi ha un error en la càrrega de recursos, he treballat amb unes funcions de que m'ajudaven a escriure ràpidament allò que necessitava de forma còmode i senzilla. A continuació podem veure'n alguns exemples.

```
template <class T>
void printError(T t){
    std::cout << "␣Resources::␣Error␣loading...␣" << t << std::endl;
}

//Escriu tots els parametres passats a la funcio log separats.
template <class... Args>
void log(const Args&... args) {
    std::cout << "[";
    internalPrintAll(args...);
    std::cout << "]" << std::endl;
}
template <class... Args>
void m_log(const Args&... args) {
    std::cout << "[";
    internalPrintAll(args...);
    std::cout << "]" << std::endl;
}
```

Capítol 4

Planificació

4.1 Objectius

A continuació llistaré els objectius generals que em vaig plantejar inicialment, tant en la definició del joc com en el seu desenvolupament, i quins han estat els resultats un cop finalitzat.

1. Trobar un estil gràfic en el qual emmarcar el projecte que resultés prou innovador i atractiu als usuaris. En un videojoc, la part gràfica és de vital importància. Per a la majoria dels usuaris, valorar la part tècnica és complicat, i també ho és valorar la part de disseny, degut a que generalment no es basa en patrons concrets. En canvi, els usuaris tendeixen a opinar sobre la part gràfica, i aquest sol ser de fet el principal aspecte pel que es reconeix un joc. Per aquest motiu, es un aspecte que cal treballar, mirant de proposar un contingut que sigui de qualitat i diferenciable d'altres productes existents.

   Resum: Trobar un disseny gràfic propi i agradable per els usuaris.

   Resultats: La combinació final dels colors plans, la texturització i el píxel art ha donat un bon resultat. El canvi d'estils dona entitat als diferents conceptes d'escena. Cada estil artístic representa un concepte: la part interactuable, el fons i la decoració. La progressió de colors al llarg del joc i la paleta escollida, genera un ambient de cova que contextualitza l'acció i posa al jugador en la situació del personatge. Els elements decoratius en píxel art són importants per fer més amena la part dels tutorials, i el fet de fer servir un estil diferent per a ells no genera confusió, i queda clar que no són elements interactuables.

2. Aconseguir uns nivells ben dissenyats per la experiència del jugador. Dissenyar primer unes mecàniques interessants, i després aconseguir que l'usuari aprengui a utilitzar-les, i pugui avançar en els nivells d'una forma progressiva, tot augmentant la dificultat del joc i sense perdre l'interès per l'excés o falta de dificultat, ni per la falta de noves experiències i reptes.

   Resum: Dissenyar unes bones mecàniques, i uns nivells que augmentin en dificultat conforme el joc avança, sense que els jugadors perdin l'interès.

   Resultats: Després de l'experiència pròpia i la observació dels usuaris, es pot afirmar que el disseny progressa adequadament, augmentant la dificultat al llarg del joc i introduint els diferents enemics de forma gradual a mesura que es van superant els nivells.

3. Fer una bona implementació. Aquest darrer punt és molt important en el projecte. Pel context d'aquest treball, entendrem que una bona implementació d'un videojoc implica:

    a) Fer un codi estructurat pel qual es pugui navegar i sigui possible fer-ne modificacions fàcilment.

    b) Que sigui extensible i escalable. Que a partir del projecte de base sigui fàcil ampliar-lo. Per exemple, poder-hi posar més enemics, més pantalles, més gràfics o altres característiques.

    c) Que funcioni amb fluïdesa, i correctament amb independència de l'ordinador d'execució, tant des del punt de vista del còmput (processador, gràfica, etc.) com de la mida o proporcions de la pantalla.

    Resum: Mantenir una bona estructura de codi, fer-lo assegurant que es pugui mantenir i ampliar, i tenir en compte els diferents dispositius en els que es podrà jugar.

    Resultats: Al llarg del projecte ha calgut ser metòdic en la estructura seguint les bones pràctiques de disseny del codi. Ha calgut triar curosament els algorismes utilitzats per no causar un excés de càlcul en alguns punts del joc. Per tal de no afectar a la jugabilitat, ha calgut identificar els llocs adequats per executar les operacions costoses, com per exemple, a l'inici o final dels nivells. Per mantenir la fluïdesa s'han assegurat uns controls agradables i unes físiques bastant realistes, i s'ha treballat usant conceptes com deltatime o frame rate fixe de 60 frames [2], [1] (utilitzant els dos per la implementació del framerate fixe que utilitza 'sleeps' que no són exactes) per aconseguir execucions correctes. Per adaptar el joc a diferents pantalles, s'ha utilitzat un sistema de vistes que s'ajusta al costat més petit de la pantalla, mantenint la proporció original. El tros de la pantalla restant es pinta en negre per mantenir l'efecte de cova.

En general, aplicant totes les mesures explicades, els objectius d'aquest apartat han quedat satisfets.

4.2 Abast del projecte

Els videojocs són projectes molt complexos que requereixen de moltes aptituds diferents, i que impliquen una dedicació a molts nivells. En aquest treball, per ser un projecte de la especialitat de computació, es destaca especialment la part d'implementació.

Dins la part de disseny, s'ha creat l'experiència, buscat l'estil gràfic, pensat les mecàniques, integrat gràfics, afegit música i sons, treballat en el disseny de nivells i de mapes, dissenyat tutorials, analitzat feedback i iterat sobre els dissenys previs.

En l'implementació, s'han tocat des del motor de joc, a com treballar amb imatges i utilitzar-les dins el joc, transformacions sobre aquestes, diferents moviments i conceptes físics per mantenir la gravetat o el moviment amb el ganxo, els diferents moviments dels enemics, els sistemes de col·lisions tant amb l'entorn com entre els actors del joc (Jugador, enemics i bales), lectura i escriptura de fitxers per tractar els nivells i les diferents descripcions de les coves, interpretació d'els documents de descripció per passar de text a objectes del codi, mantenir diferents idiomes en els textos, tractar les transicions entre les diferents escenes del joc, efectes i interpretació de spritesheets, shaders, tractament de recursos, de entrada de informació per part del jugador, integrat els sons, i treballat en general per complir els requisits de disseny desitjats.

En definitiva, podem veure que dins el projecte s'ha tocat tot el que és en sí el disseny i creació del videojoc.

Per aquest projecte, he utilitzat SFML (Simple and Fast Multimedia Library) [10], que és una llibreria multimèdia que abstreu els conceptes més bàsics de OpenGl, o d'events de sistema i m'ofereix un gestor de finestres, i algunes classes senzilles sobre les que treballar de forma més genèrica, així que la part de més baix nivell quedarà fora del que és estrictament el meu treball.

A més a més, no entraré en dissenys de marketing ni de estratègies de monetització. Per una banda perquè requereix un temps i uns coneixements dels que no disposo, però per altre, perquè no vull allunyar-me més del que és el projecte, que és un treball de final de grau per la carrera de informàtica amb especialitat de computació.

4.3 Anàlisi de la planificació inicial

La planificació temporal d'aquest projecte ha tingut la dificultat de que no sabia des del principi amb quant de temps comptava, degut a que em trobava a mig procés d'entrevistes per un lloc de treball, i aconseguir-lo o no faria canviar la meva diponibilitat, fent que la entrega del projecte fos a principis d'estiu o a l'inici del següent quatrimestre. Al final vaig entrar a treballar a l'Abril tal com havia previst, i gràcies a això la planificació va ser encertada i vaig poder seguir-la sense problemes.

Dins la planificació general, vaig fer subplans en diferents apartats que s'han mantingut. A continuació explicaré com ha evolucionat cadascun d'ells.

Idea i Concepte

Inicialment vaig estar treballant em la Preparació de la idea, concepte i tipus de joc, i per bé que al llarg de tot el desenvolupament aquests conceptes inicials han estat retocats i refinats, és sensat acceptar que la estimació de temps va resultar encertada.

Diseny del joc

A continuació el disseny de la experiència i estil desitjat. Aquesta part va ser una part de recerca molt interessant, ja que no només vaig acabar decidint les mecàniques i estil gràfic del joc si no que a més vaig aprendre un munt de coses relacionades i no relacionades amb el projecte. Seguint la planificació que havia fet vaig dedicar prop de 60 hores a aquest tasca. Era una tasca important, així que vaig esforçar-me a analitzar en detall tots els casos, iterar sobre les idees i millorar-les. Per aconseguir-ho, vaig utilitzar, no només el meu coneixement, si no també la opinió de companys desenvolupadors, i d'usuaris de diferents orígens i motivacions. El fet de treballar amb persones externes en els primers moments de la producció va ser una petita desviació de la planificació, ja que comptava que només hauria de fer falta a la part final de valoració, reiteració i testeig, però va resultar ser de gran valor també en aquest estat inicial.

Programació

El següent pas, una vegada feta la planificació del producte, era començar la programació. En vistes a la planificació inicial no hi ha hagut grans canvis. És complicat saber si la planificació per apartats concrets s'ha complert estrictament. Una vegada implementada una part, aquesta pot ser arreglada canviada o modificada en moments futurs, o retirada del codi. Així doncs, podem dir que la planificació ha estat correcte en el concepte general, tot i que en la part interna puguin no haver-se seguit els apartats concrets.

Testeig i millora

I finalment, com a darrer pas. Testeig, anàlisi, millora i redisseny. Tal com havia planificat aquest pas ha constat de dues parts, una part inicial pròpia i una part amb agents exteriors. Aquí sí que podem veure alguna desviació respecte a la planificació inicial. Arribat al punt de presentar públicament el projecte, arriba el que potser es el moment més complicat per un desenvolupador. Has creat un joc, un joc que funciona, i del que estàs molt orgullós, però i si a la resta de la gent

no li agrada? I si no entenen el que volies dir? I si la experiència que has dissenyat i programat per ells no és la que acaben rebent? Milions de preguntes la resposta a les quals poden destrossar el teu projecte. És un pas terrorífic i és més difícil del que un podria pensar des de l'experiència d'usuari. Per aquest motiu, vaig fer un canvi de planificació per tal de dedicar més hores al testeig personal per després presentar-lo al públic més polit i ben acabat. Un cop ben estudiat per mi, i amb diversos canvis i millores, vaig decidir-me a donar el pas, posar una pestanya nova al web, i pujar-hi un executable per a poder-hi jugar... i va ser fantàstic! Una vegada passat aquest primer tràngol, em penedeixo de no haver-lo passat abans. La resposta de la gent ha estat fantàstica. Per una banda molta gent m'ha felicitat i ha dit que li havia agradat molt el joc, la mecànica, els enemics, l'art ... Però encara molt més important, també m'han dit el que els hi ha faltat, què hi afegirien, què han trobat menys interessant... També algun error que a mi se m'havia passat per alt (especialment en l'executable de windows i en els textos en anglès). En general tot el feedback rebut m'ha ajudat molt, i ha fet i de fet està fent encara que el joc sigui millor, i que jo hagi crescut com a programador i dissenyador de videojocs.

En la darrera part és on s'ha acumulat més error en les estimacions. Segurament per ser la part en la que tenia menys experiència. Però l'aprenentatge ha estat molt important. M'he adonat que en realitat el testeig m'ha ocupat menys temps del que preveia. la primera part va estar ben estimada tot i després allargar-ne el temps. En la segona , en canvi, en el testeig amb agents externs, al ser ells els que juguen, el meu temps en hores ha estat molt més reduït, i al poder fer el testeig en múltiples persones al mateix temps, veig que segurament podré reduir el temps estimat.

4.4 Taula de tasques

Taula de Tasques

A continuació, a la (Figura 4.1) veiem una taula amb les diferents tasques planificades i les hores esperades per la resolució de les mateixes. El responsable de totes elles vaig ser jo mateix, ja que sóc l'únic responsable del projecte. Les dependències entre elles són essencialment l'ordre en el que apareixen, ja que és un projecte que s'ha anat ampliant a partir dels passos anteriors. Al diagrama de Gant annexat al final d'aquesta memòria es podrà veure més detalladament.

Disseny i Programació d'un Videojoc	418 hrs
▼ Preparació de la Idea, concepte i tipus	30 hrs
Definició del tipus de joc i mecàni...	30
▼ Disseny de la experiència i estil	60 hrs
Experiència de joc i estil estètic	60
▼ Programació	168 hrs
Programació	0
Motor de Joc	30
Lectura de Nivells	7
Gestor de Musica i Sons	4
Gestor de Recursos	4
Útils	8
Background	4
Player	10
Controls i interacció	10
Enemics	10
Generador de enemics i lògica	4
Menú Inici	25
Menú Pausa	7
Guardat i recuperació de dades	12
Efectes visuals	20
Integració de sons i música	5
Port a Windows	8
Millores gràfiques	0
▼ Testeig i reajustament del Disseny	160 hrs
Testeig Personal	80
Testeig extern	80

Figura 4.1: Taula de tasques

4.5 Possibles problemes i solucions

Al iniciar el projecte, vaig fer una previsió de possibles problemes amb els que em podia trobar, i vaig proposar precaucions i solucions per a ells. Una vegada acabat el projecte puc dir que no ha calgut aplicar cap de les mesures previstes, ja que no m'he trobat en cap problema ni preparat ni imprevist.

Desenvolupament

Al llarg del desenvolupament, s'han donat casos de problemes de programació. Per resoldre'ls he hagut de dedicar llargues hores de feina i recerca, però no han excedit l'estimació inicial, en la qual es preveia que la programació tindria aquestes parades.

Material

En quant a material, he pogut treballar sense problemes amb l'ordinador del que disposava a l'inici del projecte durant tot el període de treball del projecte. Gràcies a això, no he hagut de recórrer a les mesures preparades en cas de mal-funcionament o inaccessibilitat.

Persistència de dades

Tampoc s'han donat problemes de pèrdua de dades, que és un altre possible problema que havia previst i preparat. Això ha estat així principalment, perquè he treballat durant tot el procés amb eines i pràctiques que m'asseguraven la persistència i accessibilitat dels documents.

Personal

A més a més, tampoc he tingut problemes personals de cap mena, físics o temporals que m'hagin impedit treballar i seguir amb la planificació prevista.

4.6 Identificació i Estimació dels costos

Eines i materials

Aquest treball no necessita la construcció de elements materials ni una forma física de distribució, així que estalviem una part del que acostuma a ser els costos d'un projecte d'enginyeria. Donat que utilitzo l'ordinador de forma habitual, el desenvolupament d'aquest joc no ha requerit més costos que el meu dia a dia normal. Per treballar, només he utilitzat eines gratuïtes, així que no ha calgut incloure llicències en el càlcul de despeses.Cap etapa del projecte ha requerit de costos addicionals, així que la planificació inicial ha acabat resultant vàlida al arribar al final del projecte.

Tot el material requerit era un ordinador, del que ja disposava prèviament. També disposava d'alternatives per afrontar qualsevol problema des de pèrdua, robatori o malfuncionament. Per una banda un ordinador vell, que per bé que més lent, hauria servit per la programació. Per altra banda, també hauria pogut utilitzar els ordinadors de la facultat, de qualsevol de les biblioteques del sistema de biblioteques públiques de Catalunya que disposi d'ordinadors, o els ordinadors de la associació de videojocs de la universitat (VGAFIB) de la qual en sóc membre. En qualsevol cas, per si hagués hagut de comprar un ordinador nou per el projecte, a la planificació vaig comptar amb 300€ per comprar un portàtil.

Espai físic

Aquest projecte s'ha realitzat sense la necessitat d'un espai deicat exclusivament a ell. No ha fet falta una oficina ni res similar que s'hagi de afegir als costos, ja que tot el treball l'he fet des de casa. Al iniciar el projecte vivia en un pis de lloguer compartit a Barcelona, que suposava una despesa de 280€ mensuals amb aigua, llum, internet i gas inclosos. No hi ha hagut cap diferència en aquest aspecte. Però de totes formes en aquest zona hi ha altres pisos amb preus similars, i al estar aprop de la universitat, és senzill trobar companys. Per això vaig considerar vàlid aquest preu en la estimació, ja que encara que hi hagués algun problema podria trobar un lloc similar fàcilment. Un eventual canvi de pis podria haver comportar la pèrdua d'hores de feina, però atès a que la planificació preveia possibles problemes temporals, no va afectar al càlcul de costos.

Càlcul de costos

En la planificació es preveia que el projecte es realitzaria entre Febrer i Setembre, tal com ha acabat passat. En total són 8 mesos, que es poden traduïr a 2240€ en habitatge i serveis bàsics. El preu del lloguer s'ha mantingut, així que tampoc ha causat variacions.

Per la alimentació la planificació era de uns 280€ en total, considerant que gastava uns 35€ setmanals de alimentació de mitjana.

En total la planificació de costos tenia un mínim de 2520€ amb uns 300€ reservats per si feia falta un ordinador nou.

Hipòtesi de contractació professional

En cas de que aquest projecte volgués ser plantejat sobre un equip de professionals, s'hauria de considerar un dissenyador de joc i un programador. Respecte a l'art gràfic i d'àudio s'hauria de

plantejar utilitzar recursos gratuïts, comprar-ne, o contractar artistes.

Podem comptar que les primeres tasques de disseny han portat unes 90 hores, i que la part de programació unes 168. Respecte a la part del testeig i iteracions posteriors, els dos haurien de treballar junts, al ser jo sol he comptat unes 160 hores, però en aquest suposat cas repartirem les hores equitativament entre els dos.

Això acabaría amb un total de 170 hores de disseny, i 248 de programador.

170 hores són 22 dies de 8 hores de treball. Sumant els corresponents caps de setmana, tenim un total d'un mes i dos dies. El salari mig de un game designer professional a nivell europeu volta els 34.000€ de mitjana, així que hauriem de disposar d'uns 2.700€. [15] 248 hores són 31 dies de 8 hores de treball. Sumant els corresponents caps de setmana dona un mes i mig. El salari mig de un programador de C++ a Catalunya volta els 33.000€ devsalary, així que hauríem de disposar d'uns 4.050€.

Aquests salaris estan contats suposant sous mitjans de professionals, però es podria contractar gent amb menys experiència per un preu inferior.

Amb les dades aportades als apartats anteriors, veiem que el projecte podria passar a ser de ser d'uns modestos 2.820€ en el pitjor dels casos (els 300€ de l'hipotètic ordinador nou) a uns 6.7500€.

Per sort he pogut acabar tot el projecte sense haver de recórrer ni a agents externs ni a despeses imprevistes. Així doncs, el cost del projecte ha acabat resultant en uns 2520€ com estava previst.

Capítol 5

Metodología i rigor

5.1 Metodología de desenvolupament

Per fer un projecte d'aquestes característiques és necessària la integració en el mateix de moltes disciplines. En aquest apartat ens centrarem en el cas concret de la programació. Al llarg de la creació del joc, han sorgit diversos problemes importants que no tenien una solució concreta i única. Tots ells podien ser resolts de múltiples maneres, i cap d'elles millor que totes les altres, perquè hi havia molts factors a valorar i s'havia de triar quin era el que tenia més importància en cada situació.

Així doncs, he intentat sempre utilitzar uns certs criteris que veurem a continuació per tal de donar per vàlida la solució triada en cada circumstància.

1. **Escalabilitat:**

   Per un videojoc, al qual després de la primera iteració cal afegir més nivells, enemics, atacs... És important mantenir aquest concepte en ment per no haver de refer tot el codi per qualsevol millora.

2. **Implementació/usabilitat:**

   Aquest és sempre un dilema en la vida del programador. Haver d'escollir entre fer solucions fàcilment programables o de més bon utilitzar. La primera acostuma a ser una bona opció per coses finals, de poca envergadura, i que no s'hagin de re-aprofitar, sempre i quant l'estalvi de fer-ho de una forma més usable sigui significatiu. La segona és especialment important parts del codi que s'han de retocar canviar i modificar freqüentment o s'hagin d'utilitzar en múltiples ocasions i possiblement re-aprofitar en altres projectes, com per exemple l'estructura i lectura dels nivells o configuracions o el sistema d'àudio.

3. **Eficiència:**

   Aquesta part també entra sovint en conflicte amb la rapidesa de programació, així que per bé que el criteri principal és tenir una màxima eficiència, sempre es pot valorar segons la situació. Factors importants són: L'entorn del codi (s'executarà molt freqüentment o un cop per partida), la facilitat de implementació de les diverses opcions (una solució a un

mateix problema pot resoldre's en cinc minuts o en diversos dies de treball), o la capacitat del compilador de generar un codi millor en temps de compilació (en aquest cas no cal perdre el temps en millores innecessàries). En base a aquests factors, ha calgut decidir quina és la opció més adequada per aplicar a cada moment.

4. **Re-usabilitat i manteniment:**

   He utilitzat diversos sistemes dins el joc que funcionen de forma significativament autònoma. La lògica del joc o els parsers dels diferents nivells estan enfocats per funcionar per aquest joc en concret, però he utilitzat molts elements genèrics i funcionals que poden funcionar perfectament en altres projectes. Com a exemple, els encarregats dels recursos gràfics, els sons i la música, el tractament de entrada de l'usuari, o fins i tot el sistema d'escenes o d'enemics, que estan utilitzen una base que ens permet en qualsevol moment afegir variacions d'aquests sense problema.

També és cert, i això és una cosa comú a la majoria de projectes, que a la primera part del desenvolupament, sempre s'està més atent a aquests conceptes. Però arribat a cert punt, en el qual només cal afegir petits canvis o millores, treballant sobre uns sistemes ja ben construïts, és més fàcil relaxar-se, i treballar amb una visió més quantitativa que qualitativa.

5.2 Validació

Per validar el resultat d'un joc, per una part cal preveure i entendre el comporatament d'un usuari inexpert que no tingui coneixement previ. Així, s'intenta assegurar que s'indica clarament les mecàniques i reptes dels nivells, i s'intenta veure que la progressió de dificultat és correcte. Amb aquests criteris en ment, es dissenya l'experiència i s'intenten corregir els errors. Una vegada acabat el projecte, en la mesura del possible, cal trobar persones per probar el joc, i observar les seves accions i reaccions. Al fer probes amb usuaris és important no interferir en el seu comportament. Si necessiten alguna explicació, és un punt a anotar per corregir en el joc, ja que significarà que no està presentat de forma adequada. Per ajudar a aquesta validació, també cal demanar als usuaris la seva opinió, i encoratjar-los a explicar, què hi han trobat a faltar, que hi afegirien, què els ha agradat més o d'altres preguntes similars per poder acabar d'analitzar l'experiència i iterar.

Per altra banda, cal assegurar que no hi hagi problemes en la execució del joc. Aquest punt és prèvi a l'anterior en el qual es presenta el joc als uusaris, ja que els problemes d'execució generen una mala experiència, però també centren l'atenció del jugador en allò, i és més complicat extreure'n informació valuosa respecte altres aspectes del joc que queden eclipsats per aquest error. Per buscar aquests errors s'han hagut de fer proves intentant trobar tots els casos extrems o inhabituals en diferents situacions, buscar la forma de reproduïr-los, i una vegada trobats i analitzats, aconseguir-los arreglar.

Capítol 6

Sostenibilitat i compromís social

6.1 Sostenibilitat del projecte

A continuació Analitzarem la sostenibilitat del projecte en tres dimensions diferents: Econòmica, social, i ambiental.

Dimensió econòmica:

Com ja s'ha explicat i justificat, els costos del projecte tant a un nivell material com humà, han estat els que requereix el meu dia a dia. Per precaució, s'han evaluat els problemes que es podrien donar i que podrien causar despeses èxtres, però cap d'ells és significatiu o requereix de una solució econòmica important, així que molt rarament hauríen pogut alterar el pressupost.

Actualment tinc les despeses bàsiques cobertes en condició d'estudiant, així que el projecte és plenament viable. De totes formes aquesta planificació es basa en que disposo d'aquest pressupost, i no tinc la necessitat de generar benefici. Una vegada acabat el projecte, per tal de ser sostenible per si mateix s'hauria de fer un treball extra de publicitat, promoció i venda. Dins el projecte només ens centrarem en la primera part, ja que la segona requeriria tot un altre treball no menys extens o menys complex, i d'àrees molt allunyades al marc d'aquest projecte.

Aquest treball es podria haver dut a terme amb molt menys temps del que s'ha utilitzat. Per una banda perquè l'equip serà només de una persona (jo), i per l'altre perquè l'he d'alternar amb una feina extra universitària.

Això fa que el cost temporal sigui més elevat, però per una condició de pura disponibilitat horària.

La música del joc ha estat realitzada per Adrià Serrano, que ha participat de forma voluntària i per formar part d'aquest projecte i adquirir experiència en aquest camp, així que per bé que ha comportat una mica més de temps no presenta un canvi en l'impacte econòmic.

Per acabar d'assegurar la sostenibilitat econòmica, només cal veure que realment una vegada acabat aquest projecte, no necessitarà cap manteniment ni despesa extraordinària ni temporal ni fixa.

Dimensió Social:

Barcelona té múltiples empreses de videojocs, des de grans pesos en l'industria com King o Social point, a estudis que treballen per PlayStation o Nintendo, estudis més petits amb projectes més modestos, indie developers o startups que arrenquen els seus negocis al voltant dels videojocs. El cas és que aquest sector està en creixement, i especialment a la ciutat de Barcelona en gran avanç tant per les ajudes en la creació de noves empreses, moltes d'elles del sector, com per la política de potenciació de projectes relacionats amb els videojocs des del mateix govern de la generalitat, des del moment en que han estat considerats obres culturals. De totes formes, no és una competència directe, ja que el mercat dels videojocs és global i abasta tots els punts als que arribi internet i es disposi d'un ordinador. Crec, doncs, que la meva actitud pot ajudar a seguir fent créixer aquest sector i ajudar a que Barcelona segueixi sigui un referent en aquest camp. Podríem acceptar que la necessitat d'aquest producte no és de vital importància, però com les diverses obres d'art o actes culturals, aquest videojoc donarà una nova experiència que contribuirà al benestar general de la societat i a l'enriquiment cultural del país.

Dimensió ambiental:

Per aquest treball, com ja s'ha dit en múltiples ocasions al llarg d'aquesta secció, s'utilitzarà només els recursos necessaris pel meu dia a dia. Potser un lleuger augment del corrent elèctric, però donat el cas que utilitzo l'ordinador de forma habitual tant per oci com per aprenentatge o feina, podem assumir l'increment negligible. Respecte a consum o impacte comparat entre l'existència o no del meu projecte, depèn del temps emparat pels usuaris en jugant a aquest videojoc, però en general les hores de oci a l'ordinador no dependran d'aquest joc exclusivament i serien utilitzades en altres productes causant el mateix consum. A més a més, el projecte serà distribuït de forma electrònica, sense generar residus materials. Així doncs, podem admetre que no hi haurà una diferència significativa en cap aspecte ambiental.

6.2 Matriu de sostenibilitat

A la següent figura podem veure la matriu de sostenibilitat del projecte associada al projecte desenvolupat. Com hem vist al llarg d'aquest apartat i com es pot comprovar a la taula, és un projecte plenament sostenible en tots els aspectes analitzats.

	PPP	**Vida Útil**	**Riscs**
Ambiental	Consum del disseny	Impacte ecològic	Riscs ambientals
	10	18	0
Econòmic	Factura	Pla de viabilitat	Riscos econòmics
	10	18	-2
Social	Impacte personal	Impacte social	Riscs socials
	10	18	0
Rang de sostenibilitat	30	54	-2
	84		

Figura 6.1: Matriu de sostenibilitat

Apèndix A

Diagrama de Gant

Podem veure en les figures següents el diagrama de Gant utilitzat com a planificació i guia del projecte.

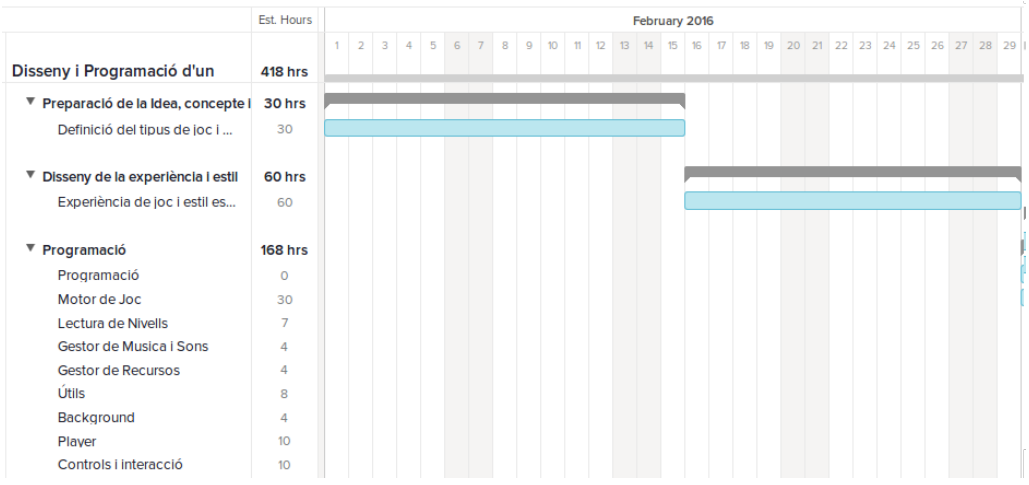

Figura A.1: Secció del Diagrama de Gant

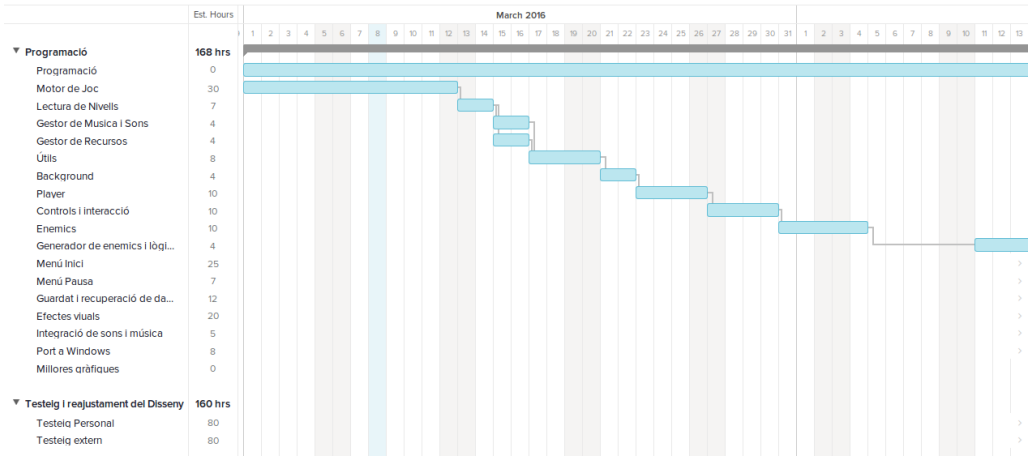

Figura A.2: Secció del Diagrama de Gant

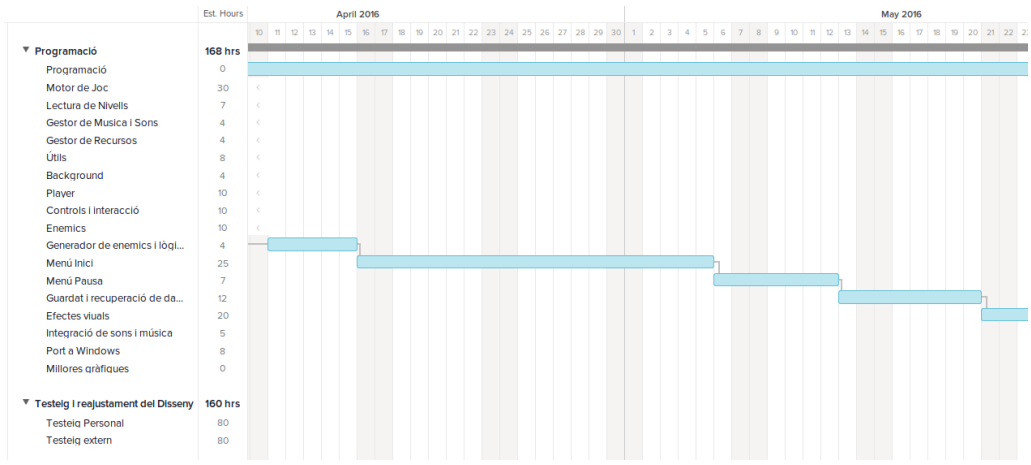

Figura A.3: Secció del Diagrama de Gant

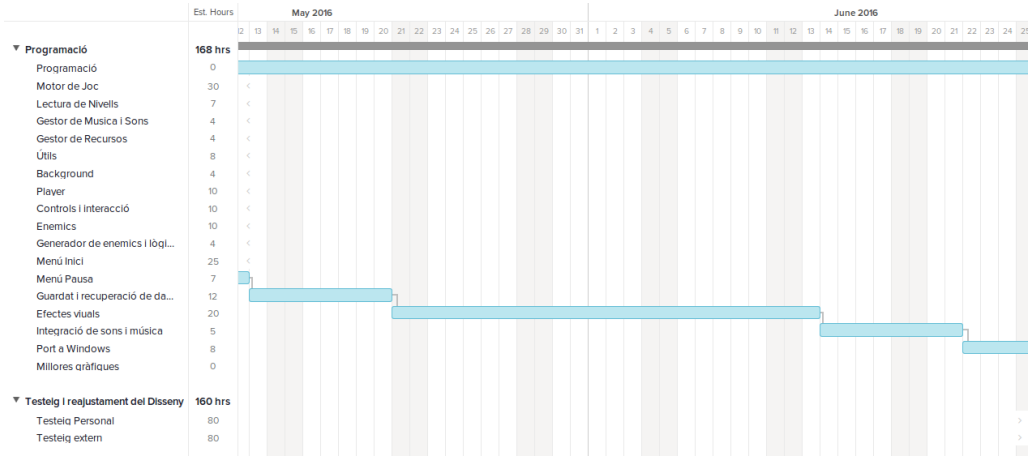

Figura A.4: Secció del Diagrama de Gant

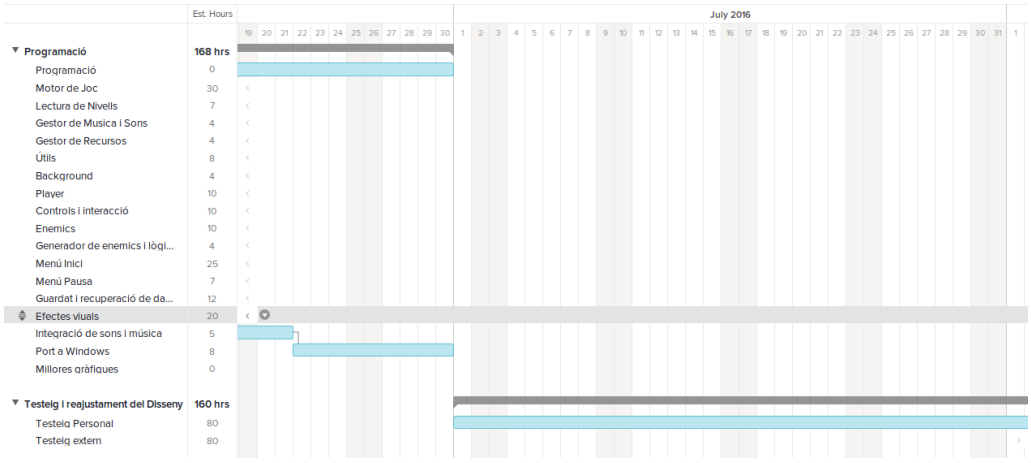

Figura A.5: Secció del Diagrama de Gant

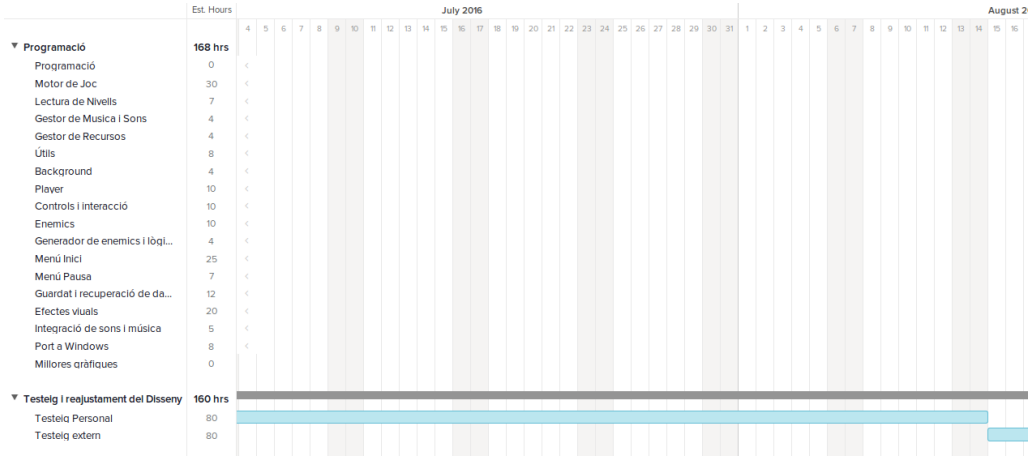

Figura A.6: Secció del Diagrama de Gant

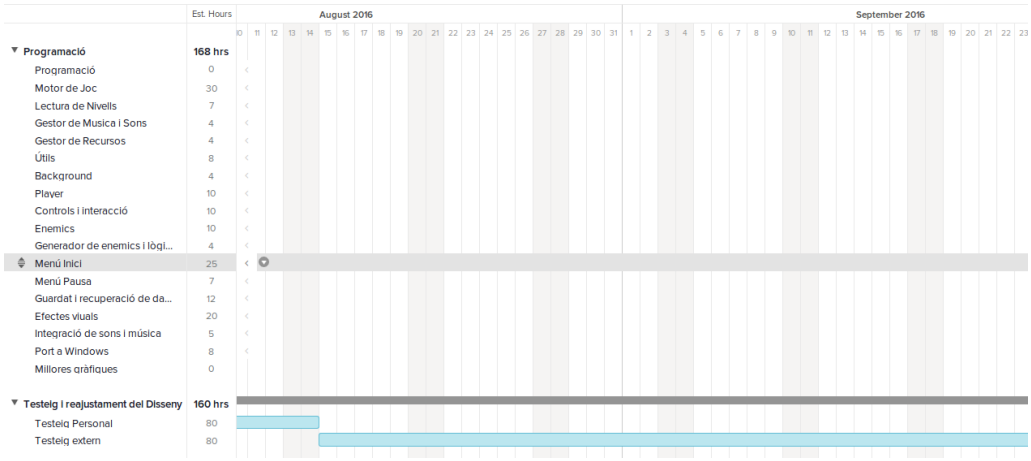

Figura A.7: Secció del Diagrama de Gant

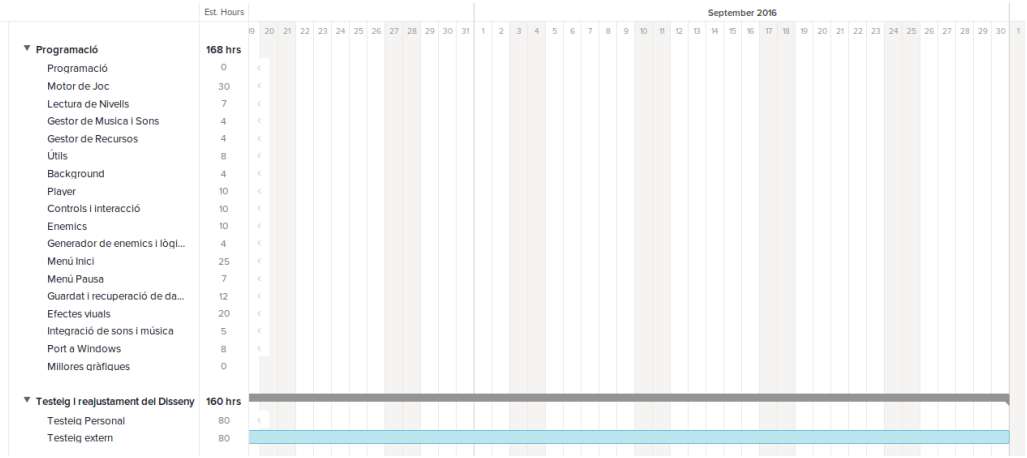

Figura A.8: Secció del Diagrama de Gant

Bibliografia

[1] Apple. What is Frame Rate, Apple Final Cut Pro 7 user manual. Accessed: 25-09-2016.

[2] Ashley. Delta-time and framerate independence. Accessed: 25-09-2016.

[3] Magnus Auvinen. “Pàgina principal de Tteeeeworlds”. Accessed: 21-03-2016.

[4] Mike Bithell. Pàgina personal de Mike Bithell. Accessed: 21-03-2016.

[5] Disney. Gravity falls oficial games webpage. Accessed: 11-10-2016.

[6] Angry Mob Games. Pàgina principal d’Angry Mob Games. Accessed: 21-03-2016.

[7] Epic Games. Pàgina principal d’Unreal engine. Accessed: 15-03-2016.

[8] Github. Pàgina principal de GitHub. Accessed: 25-03-2016.

[9] GOG. "Pàgina principal del distribuïdor de jocs GOG". Accessed: 12-03-2016.

[10] Lauren Gomila. Pàgina principal d’SFML. Accessed: 19-03-2016.

[11] Lauren Gomila. LLenguatges de programació per SFML. Accessed: 14-09-2016.

[12] Google. "Pàgina principal del cercador de google". Accessed: 10-03-2016.

[13] Disney Gravity Falls. Video de crèdits de la sèrie. Accessed: 11-10-2016.

[14] Kronos Group. Pàgina principal d’OpenGL. Accessed: 21-03-2016.

[15] Miguel Guevara. Lo que gana un desarrollador de videojuegos. Accessed: 25-09-2016.

[16] Computer Hope. Explicació relacionada amb splash screens. Accessed: 14-10-2016.

[17] Defold (King). Pàgina principal de Defold. Accessed: 15-03-2016.

[18] Nintendo. "Nintendo list of games". Accessed: 12-03-2016.

[19] Collaboration of fans. Wiki made by fans about Gravity Falls. Accessed: 11-10-2016.

[20] Nystrom Robert. Game Loop patern from the Game Programming Patterns digital book. Accessed: 10-10-2016.

[21] Sony. "Play station games". Accessed: 12-03-2016.

[22] Steam. "Pàgina principal del distribuïdor de jocs Steam". Accessed: 12-03-2016.

[23] Unity Technologies. Pàgina principal de Unity. Accessed: 15-03-2016.

[24] Trello. Pàgina principal de Trello. Accessed: 25-03-2016.

[25] Julian Winter. Pàgina principal de SDL. Accessed: 19-03-2016.

[26] Albaladejo Xavier. Conoce scrum. Accessed: 25-03-2016.